EPFL

Thèse n° 9669

# Sparse Autoencoders for Speech Modeling and Recognition

Présentée le 2 février 2023

Faculté des sciences et techniques de l'ingénieur
Laboratoire de l'IDIAP
Programme doctoral en génie électrique

pour l'obtention du grade de Docteur ès Sciences

par

**Selen Hande KABIL**

Acceptée sur proposition du jury

Prof. A. Ijspeert, président du jury
Prof. H. Bourlard, directeur de thèse
Prof. H. Christensen, rapporteuse
Dr M. Cernak, rapporteur
Prof. J.-Ph. Thiran, rapporteur

École polytechnique fédérale de Lausanne

2023

Our true mentor in life is science.
— Kemal Atatürk

To my parents...

# Acknowledgements

I have been incredibly fortunate and honoured to be the last student of Prof. Hervé Bourlard. Without his constant patience, encouragement and advices, this thesis would not have been possible. I sincerely thank Hervé for this incredible journey.

I thank Swiss National Science Foundation (SNSF) for funding my work through the grant 200021-175589 for the SHISSM project. I kindly thank my thesis committee members Prof. Auke Ijspeert, Prof. Jean-Philippe Thiran, Prof. Heidi Christensen, and Dr. Miloš Ćernak for their insightful comments and questions.

I would also like to thank Dr. Mathew Magimai-Doss who introduced me to Idiap and speech processing community. I was fortunate to complete my masters thesis under his supervision. And, a special mention goes to Dr. Ina Kodrasi, who has been a mentor for me. Her hardwork, dedication and consistency still amaze me.

The last 4.5 years was such an incredible journey. I learned a lot, about myself too. With all these ups and downs, I am most grateful to have two friends: Aysu and Isinsu. We survived through our PhDs together. Your existence created the little comfort zone in Lausanne that was very much needed.

Idiap introduced me incredible friends, some of them I am happy to call as my second family. I would like to thank Gülcan and PE for their support and resistance to my never-ending nagging. I thank Florian for his crazy sense of humour and kind soul and, of course, for random walks around windy Martignyland during which we questioned the life and stuff. I thank Hakan for his sarcasm and constant opposition for almost every subject matter existed. I thank Teja, Skanda and Reema for Friday night game events which were the highlight of my busy weeks. I am grateful to get to know Hannah, Pavan, Sibo, Pranay, Dhannanjay, Leslie, Bastian and Sophie in the early phases of my PhD. Their encouragement and insights really helped me believing in the light at the end of the tunnel and persevere. I would like to thank Eklavya for hosting the best parties in the history of Martigny, which were great help for getting back to normal after pandemic. I also thank Apoorv, Sargam, Zohreh, Amir, Ravi, Fabio, Andrei,

**Acknowledgements**

Louise, Enno, Julian, Sandrine, Amrutha, François and Srikanth. And, of course, I want to thank all past and current members of 303, the best office ever.

Finally, last but by no means least, I am grateful to my whole family, especially my parents for their patience, support and guidance. I love you guys, kedigiller rocks!

*Martigny, October 2022* Hande

# Abstract

Speech recognition based applications upon the advancements in artificial intelligence play an essential role to transform most aspects of modern life. However, speech recognition in real-life conditions (e.g., in the presence of overlapping speech, varying speaker characteristics) remains to be a challenge. The current state of the research to achieve robust speech recognition mostly depends on building systems driven by complex deep neural networks. Nonetheless, speech production process enables low-dimensional subspaces which can carry class-specific information in speech. In this thesis, we investigate the exploitation of this low-dimensional multi-subspace structure of speech towards the goal of improving acoustic modeling for automatic speech recognition (ASR).

This thesis mainly focuses on the sparse autoencoders for sparse modeling of speech, starting from their often-overlooked connection with sparse coding. We hypothesize that whenever speech signal is represented in a high-dimensional feature space, the true class information (regarding the speech content) is embedded in low-dimensional subspaces. The analysis on the high-dimensional sparse speech representations obtained from the sparse autoencoders demonstrates their prominent capability of modeling the underlying (e.g., sub-phonetic) components of speech. When used for recognition, the representations from sparse autoencoders yield performance improvements. Finally, we repurpose the aforementioned sparse autoencoders for pathological speech recognition task in transfer learning framework.

In this context, the contribution of this thesis is twofold: (i) in speech modeling, proposing the use of sparse autoencoders as a novel way of sparse modeling for extracting the class-specific low-dimensional subspaces in speech features, and (ii) in speech recognition, demonstrating the effectiveness of these autoencoders in the state-of-the-art ASR frameworks towards the goal of improving robust ASR, in particular on far-field speech from AMI and pathological speech from UA-Speech datasets.

**Keywords:** automatic speech recognition, deep neural network, sparse autoencoder, representation learning, sparsity

# Résumé

Les applications basées sur la reconnaissance vocale avec les progrès de l'intelligence artificielle ont un rôle essentiel dans la transformation de la plupart des aspects de la vie moderne. Cependant, la reconnaissance vocale dansles conditions réelles (par exemple, en présence de discours qui se chevauchent, de caractéristiques différentes du locuteur) restent un défi. L'état actuel de la recherche sur la parole robuste la reconnaissance dépend principalement de la construction de systèmes pilotés par des réseaux complexes de neurones profonds. Néanmoins, le processus de production de la parole permet des sous-espaces de faible dimension qui peuvent transporter informations spécifiques à la classe dans le discours. Dans cette thèse, nous étudions l'exploitation de cette structure multi-sous-espace de faible dimension de la parole dans le but d'améliorer l'acoustique modélisation pour la reconnaissance automatique de la parole (RAP).

Cette thèse se concentre principalement sur les autoencodeurs épars pour la modélisation de la parole éparse, en partant de leur connexion souvent négligée avec le codage épars. Nous faisons l'hypothèse que lorsque signal de parole est représenté dans un espace de caractéristiques à haute dimension, la véritable information de classe (concernant le contenu de la parole) est intégrée dans des sous-espaces à basse dimension. L'analyse des représentations de la parole à haute dimension obtenues à partir des autoencodeurs sparse démontre leur grande capacité à modéliser les composantes sous-jacentes (par exemple, subphonétiques) de la parole. Lorsqu'elles sont utilisées pour la reconnaissance, les représentations obtenues à partir des autoencodeurs sparse améliorent les performances. Enfin, nous ré-utilisons les auto-codeurs clairsemés ci-dessus pour la reconnaissance de la parole pathologique dans un cadre d'apprentissage par transfert.

Dans ce contexte, la contribution de cette thèse est double : (i) modélisation de la parole, nous proposons l'utilisation d'auto-encodeurs épars comme une nouvelle approche de modélisation éparse, pour extraire les sous-espaces à faible dimension spécifiques à certaines classes du signal de parole, et (ii) reconnaissance de la parole, nous démontrons l'efficacité de ces auto-encodeurs sur des modèles de RAP de pointe avec comme objectif l'amélioration de la reconnaissance vocale robuste, plus spécifiquement sur la reconnaissance vocale lointaine,

sur la base de donnée AMI et sur la parole pathologique sur la base de données UA-Speech.

**Mots-clés :** reconnaissance automatique de la parole, réseau de neurones profonds, auto-encodeur épars, apprentissage de la représentation, parcimonie

# Contents

# List of Figures

# List of Tables

# List of Acronyms

**AE** Autoencoder

**ASR** Automatic Speech Recognition

**CE** Cross Entropy

**DNN** Deep Neural Network

**HMM** Hidden Markov Model

**IHM** Individual Headset Microphone condition in AMI dataset

**KL** Kullback-Leibler

**LF-MMI** Lattice-Free Maximum Mutual Information

**MFCC** Mel-Frequency Cepstral Coefficient

**MMI** Maximum Mutual Information

**MSE** Mean Square Error

**PCA** Principal Component Analysis

**ReLU** Rectified Linear Unit

**SDM** Single Dinstant Microphone condition in AMI dataset

**SDR** Sparse Distributed Representation

**TDNN** Time-Delay Neural Network

**VAE** Variational Autoencoder

**WER** Word Error Rate

# 1 Introduction

With the advancements in artificial intelligence, including the revival of Deep Neural Networks (DNNs) due to the availability of extensive data and computational resources [LeCun et al. (2015); Bengio et al. (2021)], wide range of Automatic Speech Recognition (ASR) applications have been developed. Some of the major applications of ASR include smart personal assistants (e.g., Alexa, Google Assistant, and Siri), smart-home devices controlled with speech, systems for transcribing meetings, automatic subtitle generation for news or movies, military, and health care services.

As ASR field continues to advance, there is an increasing demand to enhance the existing technology in order to continually deal with the open challenges. For instance, the aforementioned everyday life applications require ASR to be robust to full-range of real-world noise and other acoustic distortions. However, recognizing unconstrained conversational speech, possibly corrupted with overlapped speech, is still a very challenging task, and the current state-of-the-art systems are still a long way from being able to reach human performance levels. In addition, the distortions due to speaker characteristics (e.g., pathology) introduce various challenges especially for health care applications. Hence, robustness remains to be an important research direction.

State-of-the-art hybrid (pipeline-based) ASR techniques exploit the advances in deep learning by employing a variety of complex DNNs for estimating the probability distribution over speech data with a Hidden Markov Model (HMM) based back-end for sequence modeling of speech. However, these state-of-the-art techniques still fail to exploit certain intrinsic properties of speech. Although not properly leveraged before, exploiting intrinsic structure in speech holds the key for robust ASR and also provides a thorough understanding of speech modeling for speech processing tasks in general. The research presented in this thesis addresses this gap by proposing deterministic sparse autoencoders to learn informative, easier to interpret speech representations, mainly towards improving ASR in mismatched conditions.

In this pursuit, we start from the basics and exploit the often-overlooked connection of autoencoders with Principal Component Analysis (PCA) and with sparse coding techniques.

Inspired by these two techniques, we develop autoencoders for low-rank and sparse modeling of speech data, respectively. We observe performance improvements when the representations obtained from these autoencoders are used for recognition, especially from sparse autoencoders for sparse modeling. In addition, the qualitative analysis of the learned high-dimensional sparse speech representations shows that the sparse autoencoders are able to model the low-level sub-phonetic components of speech. This helps to discover similarities and to build correlations between observed and unobserved speech samples.

With an emphasis on notions of population and life-time sparseness, we anticipate further improving sparse autoencoders with moderate computational cost. The speech representations obtained from such autoencoders are expected to be task-agnostic and hence, useful in the transfer learning on mismatched speech. Thereby, the conclusion of this thesis is also reinforced by the transfer learning framework for pathological speech recognition.

The rest of this chapter is organized as follows. Section 1.1 provides the general motivation behind the approach proposed in this work. Section 1.2 summarizes the contributions of this thesis. Finally, Section 1.3 presents the thesis outline.

## 1.1 Motivation

In this thesis, we focus on the sparse autoencoders for sparse modeling of speech, with the goal of learning informative and interpretable speech representations, mainly towards improving ASR in acoustically mismatched conditions. We hypothesize that modeling speech by utilizing its intrinsic structure holds the key for robust ASR and also provides thorough understanding of speech modeling for speech processing tasks in general. To play upon the intrinsic structure of speech signal, we consult speech production knowledge.

Human speech production process driven by cognitive planning leads to *parsimonious hierarchical* structure. Articulation is due to the coordination among the articulators such as tongue, teeth, lips, palate etc. During articulation, sub-phonetic components are produced by the specific realization of only a few highly constrained articulators at any given time. This leads to *parsimonious* nature of speech. In addition, when we go higher in the compositional *hierarchy* (Figure 1.1), to produce a word, we only use a few phonemes among ~40 phonemes in English language. Similarly, we use one word (out of ~170,000 words in English language) at a time to produce a sentence.

In other words, *parsimony* and *hierarchy* in the structure of speech go hand in hand, especially for deciphering the representation space. As shown in Figure 1.1, the higher we go in the hierarchy (i.e., from phonetic units to sentences), the higher-dimensional and sparser the representation space becomes. This leads to thinning out of meaningful data components in higher-dimensional representation space, resulting in so called class-specific low-dimensional subspaces in speech features [Stevens (2000); Jansen and Niyogi (2006)].

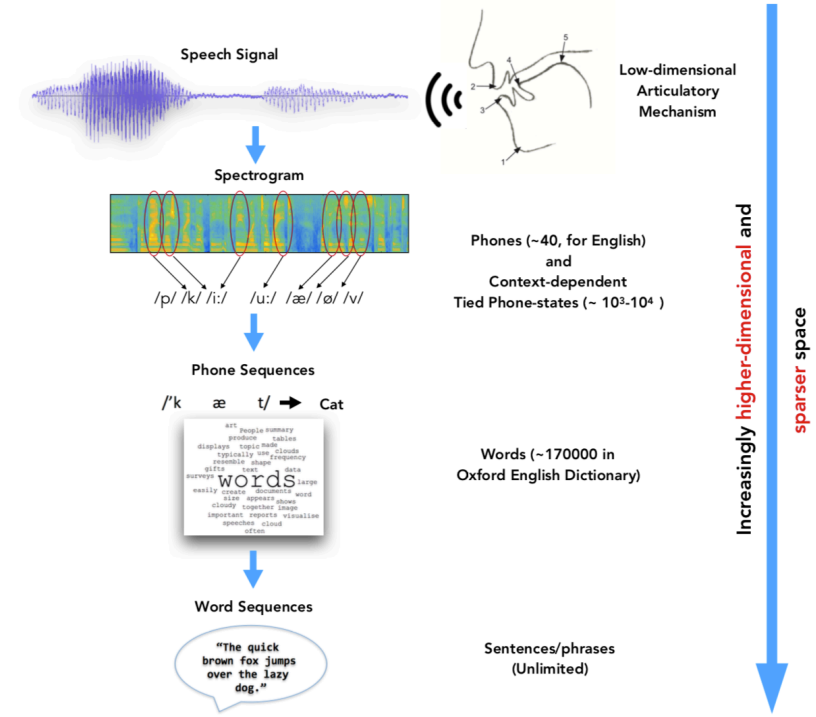



***Figure 1.1:*** *Parsimonious hierarchical structure of human speech. The higher we go in the hierarchy (i.e., from subphonetic units to sentences), the higher dimensional and sparser representation space becomes. This parsimonious hierarchical structure of speech leads to class-specific low-dimensional subspaces in speech features [Stevens (2000); Jansen and Niyogi (2006)]. In this thesis, we aim to model and exploit these internal components by means of sparse autoencoders for various speech processing tasks. Figure has been reproduced from [Dighe (2019)] with permission.*

There exist studies that advantageously utilize the parsimonious hierarchical nature of speech. For instance, to encapsulate these subspaces, sparse modeling is shown to be an efficient way and has been found to be useful when performing noise robust speech recognition [Sainath et al. (2011); Gemmeke et al. (2011)]. Similarly, [Dighe (2019)], which serves as the basis of the research in this thesis, focuses on low-rank and sparse modeling of DNN based posterior features. The study proposes to enhance DNN posteriors by projecting them onto the manifolds of the underlying classes using PCA or sparse coding based dictionaries.

Besides the compliance to the nature of human speech (depicted in Figure 1.1), another motivation for exploring sparse modeling in speech research is compliance to the human brain functioning. Findings in neuroscience and psycho-acoustics suggest that the human brain exploits sparse coding and hierarchically analyzes the stimuli at the level of cognitive processing and neural activities [Allen (1995); Olshausen and Field (1996)]. Integrating this knowledge can therefore help unravel many phenomena observed in speech and design transparent models

that are robust to noise and other acoustically mismatched conditions [Deng et al. (1997); Frankel and King (2001); King et al. (2007)]. Hence, in this thesis, given the parsimonious hierarchical nature of speech, we devise sparse autoencoders for learning informative and interpretable speech representations in unsupervised and computationally efficient manner towards robust ASR.

## 1.2 Contributions

We examine the often-overlooked connection of autoencoders with PCA (Section 3.1.1) and with sparse coding (Section 3.2) for low-rank and sparse modeling of speech data respectively, with the motivation and the reference point (i.e., [Dighe (2019)]) presented in Section 1.1. Our focus is mainly on sparse modeling by means of sparse autoencoders; yet, for the sake of completeness, we also investigate the low-rank modeling approach by means of undercomplete autoencoders (Section 3.1.1).

In **Chapter 4**, we propose the use of sparse autoencoders (Section 3.2) for sparse modeling of likelihoods from the acoustic model and observe improvements in recognition performance. In addition, the analysis conducted on high-dimensional sparse features obtained from sparse autoencoders demonstrate that these features are capable of modeling even phone-level components in speech.

In **Chapter 5**, we examine the use of sparse autoencoders for sparse modeling of acoustic features. Acoustic features (Section 2.1.1) are known to be less stable and less robust compared to the likelihoods from the acoustic model. Sparse modeling of acoustic features by means of sparse autoencoders results in better performance improvement on far-field speech data. Therefore, in the rest of the thesis, we continue using acoustic features to train autoencoders. However, the analysis on high-dimensional sparse features obtained from sparse autoencoders are also shown to be less robust and less informative (Section 5.4), compared to their likelihood counterparts (Section 4.4). This is due to the close-to-zero values in the sparse features to obfuscate the findings from subspace analysis. Therefore, in the following chapter, we focus on other sparse autoencoder configurations, capable of producing hard-zero encodings.

In **Chapter 6**, we propose the use of implicitly constrained sparse autoencoders (elaborated in Section 3.2) with internal sparsity mechanisms, inspired from competitive learning studies [Srivastava et al. (2013)]. These internal sparsity mechanisms set the predetermined portion of the autoencoder activations to zero without introducing any additional regularizer term in the autoencoder loss function (i.e., hence the name implicitly regularized). The analysis conducted on high-dimensional sparse features obtained from sparse autoencoders demonstrates that robust modeling of underlying speech components is possible, even in the presence of additive noise. This mainly depends on the sparseness viewpoint (Section 3.2.4) adopted by the sparse autoencoder.

In **Chapter 7**, we propose the use of best performing implicitly constrained sparse autoencoder

model (from Chapter 6) for transfer learning on pathological speech. We anticipate the success of this model for projecting the acoustic space of pathological speech closer to healthy control acoustic space, given its capability for robust modeling of underlying speech components in healthy speech. Consequently, we manage to obtain promising improvement for pathological speech recognition performance with our proposed transfer learning framework, even though the baseline acoustic model is finely-tuned and therefore very sensitive.

Hence, our contributions in this thesis are the following:

**Autoencoders for low-rank and sparse modeling of speech** : [Dighe (2019)] introduces novel low-rank and sparse modeling approaches based on PCA and sparse coding for ASR. Followingly, we propose the use of a simple sparse autoencoder for sparse modeling of speech data (e.g., likelihoods from the acoustic model in Chapter 4, acoustic features in Chapter 5) for improving state-of-the-art recognition systems on close-talk and far-field speech. Compared to [Dighe (2019)], our approach is unsupervised and computationally efficient, as it requires less training steps. It also provides performance improvements over a more challenging state-of-the-art recognition system. In addition, to present a complete overview, we investigate the use of an autoencoder with a bottleneck layer for low-rank modeling of speech data (e.g., likelihoods from the acoustic model in Chapter 4, acoustic features in Chapter 5).

**Analysis of the learned speech encodings** : We model the global characteristics of the speech through compact, low-rank representations by employing an autoencoder with a bottleneck layer. Whereas, with sparse autoencoders, we can model different low-level (e.g., phone-level, articulatory-level) components of the speech data. To test this, we perform the analysis (Section 4.4 and Section 5.4) on the high-dimensional sparse features (i.e., embeddings, representations, encodings, hidden layer activations) obtained from sparse autoencoders. When these feature vectors are matched together based on a similarity score and a matching threshold, they are observed to share similar articulatory configurations.

**Implicitly constrained competitive sparse autoencoders** : We observe the potential of simple sparse autoencoders for producing meaningful representations which model underlying components in speech data. To further exploit this potential, we investigate sparse autoencoder models concerning different viewpoints to sparseness and sparsity mechanisms (Section 3.2.4). Thorough investigation shows that there exist several ways for imposing sparsity to guide the autoencoder. More precisely, an autoencoder can be guided in two ways; explicitly by means of different penalty terms in the loss function (utilized in Chapter 4 and Chapter 5) or implicitly by altering the internal model components (i.e, activations, weights). Implicitly constrained sparse autoencoders have an internal sparsity mechanism, which may adopt different sparseness viewpoints such as population sparseness and life-time sparseness. Thanks to this mechanism, the model can produce high-dimensional sparse features with hard-zero values, which simplifies tracking the patterns in subspace analysis (Section 6.3.3). To the best of our knowledge, such sparse autoencoders have not been exploited in speech processing tasks before.

**Transfer learning on mismatched data** : We select the best performing implicitly constrained sparse autoencoder model (from Chapter 6) and repurpose it for transfer learning on pathological speech in Chapter 7. The experiments are run for three scenarios: (1) as-is (with fixed acoustic model) (2) finetuning the acoustic model using reconstructed pathological speech features from sparse autoencoder (3) training the acoustic model from scratch on reconstructed pathological speech features from sparse autoencoder. With the presented transfer learning framework, we manage to obtain improvements on recognition performance, despite the sensitivity of the finely-tuned baseline acoustic models. This indicates that sparse autoencoder models can indeed be useful for learning meaningful and generalizable speech features.

## 1.3 Thesis Outline

We present the organization of this thesis by briefly describing the main goal of each of its constituent chapters.

Chapter 2, *Background on automatic speech recognition*, introduces the key components of ASR pipeline with a particular focus on the state-of-the-art acoustic modeling. In addition, the baseline system, datasets and evaluation metrics for the experiments in the following chapters are described.

Chapter 3, *Background on autoencoders*, provides essential background information about autoencoders with special emphasis on sparse autoencoders and sparsity.

Chapter 4, *Low-rank and sparse modeling of LF-MMI log-likelihoods*, revisits the prior work [Dighe (2019)], its strengths and weaknesses in particular, to motivate the proposed approach of this thesis. After the proposed framework is introduced, the experimental results for recognition performance and analysis of the learned high-dimensional sparse speech encodings are presented.

Chapter 5, *Low-rank and sparse modeling of acoustic features*, presents the experimental results for recognition performance and analysis of the learned high-dimensional sparse speech encodings while the autoencoders are fed with acoustic features.

Chapter 6 *Implicitly constrained sparse autoencoders*, explores various sparse autoencoder architectures from different sparsity viewpoints to enhance the speech representations towards improving the recognition performance and generalization power.

Chapter 7 *Transfer learning for pathological speech recognition*, introduces the transfer learning framework which repurposes the best performing sparse autoencoder model from Chapter 6 for pathological speech recognition.

Chapter 8 *Conclusion and future work*, derives the main conclusions of this thesis and provides possible directions for future work.

# 2 Background on Automatic Speech Recognition

In this chapter, we provide a brief background on Automatic Speech Recognition (ASR), with special emphasis on the conventional Hidden Markov Model (HMM)-based ASR. Section 2.1 describes the key components in the ASR pipeline, providing details for the acoustic features, DNN-based acoustic modeling and various hierarchical speech units that we exploit for modeling in this thesis. In Section 2.2, we introduce the Lattice-Free Maximum Mutual Likelihood (LF-MMI) ASR system which constitutes the configuration for the state-of-the-art baseline systems. Finally, we give details of the databases, evaluation metrics and the baseline systems in Section 2.3, Section 2.4 and Section 2.5, respectively.

## 2.1 Key Components in the ASR Pipeline

A conventional HMM based ASR system consists of individual components as shown in Figure 2.1. The speech signal is first passed through a signal processing and feature extraction component. This component enhances the signal (i.e., reduces noise and distortions) and extracts multi-dimensional acoustic features. These features are then used for acoustic modeling. The acoustic model computes the data-likelihoods given the input acoustic feature sequence by integrating knowledge about acoustics and phonetics. These acoustic data likelihoods are sent to the decoder component. A decoder can be conceptualized as an implementation of an HMM which brings multiple hierarchies of speech together in a graph. Combining the data likelihoods from the acoustic model and word sequence probabilities from the language model, this component finds the most probable path (i.e., the least expensive path) in the decoding graph to produce the hypothesized word sequence.

In this thesis, we mainly focus on the acoustic modeling component. Using different autoencoder models (elaborated in Chapter 3), we work toward learning better representation alternatives for the acoustic data likelihoods (in Chapter 4) and acoustic features (in Chapter 5) with the end goal of improving ASR performance. In essence, an autoencoder is inserted in the ASR pipeline as an additional component after the acoustic model in Chapter 4 and before the acoustic model in Chapter 5. After exploring different sparse autoencoders for this repre-

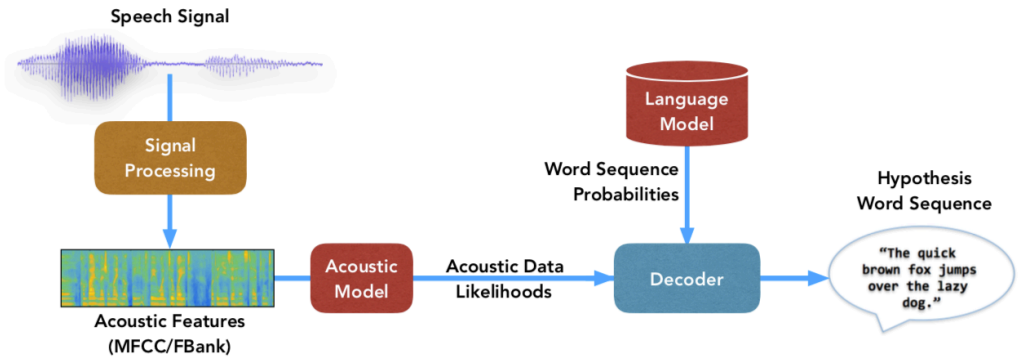



***Figure 2.1:*** *Key components of the ASR pipeline. Figure has been reproduced from [Dighe (2019)] with permission.*

sentation learning problem in Chapter 6, we specifically focus on plugging the pre-trained, off-the-shelf sparse autoencoder in different pipelines designed for different ASR tasks (i.e., pathological speech) in Chapter 7. We did not explore modifications or improvements on the language modeling component in this thesis.

Before we proceed with the details of the DNN-HMM based acoustic model training, we present some standard types of acoustic features, a short overview on the HMM, and speech units relevant to this thesis. Moreover, we introduce the baseline systems, datasets and evaluation metrics used in this thesis.

### 2.1.1 Acoustic Features

The most popular speech features used in ASR: spectrograms, Mel-filter banks, Mel-frequency cepstral coefficients. Regardless of the feature type, most of these features share some main principles, being computed by applying some specific operations. In general, they are biologically-inspired, imitating the human sound perception system.

In this thesis, we employ Mel-Frequency Cepstral Coefficients (MFCCs). As the ASR pipeline (Figure 2.1) we use for our experiments are implemented using Kaldi [Povey et al. (2011)] speech processing toolkit, MFCCs are extracted with Kaldi feature extractor following the steps below:

**Sampling and pre-emphasis:** Speech signal is captured at a fixed sampling rate, typically 8kHz or 16kHz. Sampling at a high-frequency rate can result in low-frequency components of the signal having high energy while the high-frequency components might be subdued. Hence, a pre-emphasis operation is done to the original signal $s(t)$ as follows to amplify the high-frequency components of the signal:

$$s'(t) = s(t) - \alpha s(t-1) \tag{2.1}$$

where $\alpha$, the first-order filter coefficient, is typically taken to be 0.97.

**Windowing and FFT:** In this step, the pre-emphasized time-domain signal is converted to the frequency-domain to analyze the behavior of different frequency components over time. A Fourier transform of the whole signal is not a suitable choice since the signal is not stationary and a signal-level transform would result in losing the information about the evolution of the frequency contours. To avoid these issues, the signal is analyzed in a sliding window fashion as we can assume stationarity for very short durations of time. Therefore, we apply Fast Fourier Transform (FFT) to one window at a time to get a frequency-domain representation of the short duration signal in the window. FFT of each window results in one spectral feature frame in frequency-domain. At the end, all the frames of spectral features are concatenated together adjacently to form the spectrogram of the signal. Typically, window length is set as 25msec which contains 400 samples of the pre-emphasized time-domain signal if the sampling rate is 16kHz. Before FFT, a Hamming window function is also applied to account for the fact that FFT assumes the time-domain signal to be of infinite length. The spectrogram output from FFT is converted to a power spectrum by taking square of the amplitudes in the spectrogram.

**Filterbank analysis:** In this step, Mel-scale filters are applied to the power spectrum to imitate human auditory perception which is more discriminatory for the lower frequencies, compared to the higher ones. There are typically 40 overlapping triangular filters whose centers are placed evenly in the Mel-frequency domain. Finally, we compute the log of energy under each Mel-filter. At this stage, each frame of the spectrum has 40 Mel-filterbank energies. These features are called log Mel-filterbank energies (also known as Fbank).

**Decorrelating with DCT:** The dimensions of the Fbank energy features are highly correlated. With decorrelation, we can use Gaussian modeling with diagonalized covariance and exclude high-frequency components. To do so, we take the Discrete Cosine Transform (DCT) of the Fbank features and pick the first few coefficients (usually 13) which are significant for ASR. These coefficients are called Mel-frequency cepstral coefficients (MFCCs).

Depending on the configuration of the acoustic model for training (in Kaldi), the feature extractor can produce low-resolution features (e.g., 13 MFCCs per frame to train Gaussian Mixture Models (GMMs)) or high-resolution features (e.g., 40 Mel filter banks to train neural network models).

### 2.1.2 Hidden Markov Model

Over the last several decades, Hidden Markov Models (HMMs) have served as the backbone of almost all large-scale ASR systems, thanks to their success in sequence processing. Here, we introduce the basics of HMMs.

A Hidden Markov Model is a Markov chain where each state generates an observable discrete symbol or continuous-valued vector as per a state-conditional probability distribution function. While the emitted observations are visible to an observer, the underlying Markov process is hidden. The hidden state sequence is non-deterministic and can only be statistically estimated based on the observation sequence and the parameters (emission probability density function) of the model. Here, we consider only continuous density HMMs which emit real-valued multi-dimensional vectors as observations. The random variable denoting the observed sequence is defined as $X = \langle X_1, X_2, \ldots, X_T \rangle$.

Hence, HMM can be completely defined by the following components:

- Set of states, $\mathbb{Q} = \{q_1, q_2, \ldots, q_K\}$: Random variable $Q_t$, denoting hidden state at time $t$, takes values from this set.
- Set of observations, $\mathbb{R}^m$: Random variable $X_t$, denoting the observation emitted at time $t$, takes a value $\boldsymbol{x}_t \in \mathbb{R}^m$.
- Prior probabilities, $\pi = \{\pi_1, \ldots, \pi_K\}$ where $\pi_k$ denotes the probability that the Markov chain will start with particular $q_k$ state.

$$\pi_k = P\left(Q_1 = q_k\right) \quad \text{s.t.} \quad \pi_k \geq 0 \quad \forall k, \quad \sum_{k=1}^{K} \pi_k = 1 \tag{2.2}$$

- Transition probabilities, $a_{i,j}$ denote the probability that the Markov chain will go from one particular state ($q_i$) to another ($q_j$).

$$a_{i,j} = P\left(Q_t = q_j \mid Q_{t-1} = q_i\right) \quad \text{s.t.} \quad a_{i,j} \geq 0 \quad \forall j, \quad \sum_{j=1}^{K} a_{i,j} = 1 \tag{2.3}$$

- Emission probabilities, $b_k(\boldsymbol{x})$ denote the probability of an observation $\boldsymbol{x} \in \mathbb{R}^m$ being generated when the underlying hidden state is $q_k$.

$$b_k(\boldsymbol{x}) = P\left(\boldsymbol{x} \mid q_k\right) \tag{2.4}$$

An HMM based on a first-order Markov chain makes two important assumptions. The first assumption is the first-order Markovian assumption. That is, the conditional probability distribution of the future state is only dependent on the current state.

$$P\left(Q_t \mid Q_1^{t-1}\right) = P\left(Q_t \mid Q_{t-1}\right)\big) \tag{2.5}$$

The second assumption, famously called HMM conditional-independence assumption, states that the observation emitted at time $t$ is dependent only on the hidden state at time $t$, and is conditionally independent of the past hidden states as well as observations.

$$P\left(X_t \mid X_1^{t-1}, Q_1^t\right) = P\left(X_t \mid Q_t\right) \tag{2.6}$$

Employing HMMs for any task usually leads to one or more of the following three standard problems: (1) finding the likelihood of an observation sequence given the HMM parameters, (2) finding the most likely hidden state sequence given an observation sequence and the HMM parameters, and (3) finding the parameters of the HMM given a set of observation sequences. To solve these three problems, famous HMM-based algorithms, namely Forward-backward algorithm, Viterbi algorithm, and Baum-Welch algorithm are used, respectively. We refer the reader to [Rabiner (1989)] for complete details of these algorithms.

### 2.1.3 Mathematical Formulation of HMM-based ASR

In a typical HMM based ASR framework, the hypothesized word sequence $\hat{W}$ is estimated from the sequence of acoustic features $X = \{x_1, x_2, \ldots, x_T\}$, where $x_t$ is a standard acoustic feature at time $t$, as

$$\begin{aligned} \hat{W} &= \underset{W}{\operatorname{argmax}} P(W \mid X) \\ &= \underset{W}{\operatorname{argmax}} \frac{p(X \mid W)P(W)}{p(X)} = \underset{W}{\operatorname{argmax}} p(X \mid W)P(W) \end{aligned} \tag{2.7}$$

where $p(W)$ is the probability of the word sequence $W$ estimated from a language model and $p(X \mid W)$ is the likelihood of the acoustic sequence conditioned on the word sequence, estimated from an acoustic model.

We ignore the denominator probability $p(X)$, since it is independent of the word sequence $W$ in the maximization argument. Assuming that the observation sequence $X$ is generated by a HMM, the task at hand is to compute its probability by marginalizing all possible hidden state sequences (i.e., using the Forward-Backward algorithm). Thus, $p(X \mid W)$ is computed as

$$\begin{aligned} p(X \mid W) &= \sum_Q p(X \mid Q, W)P(Q \mid W) \\ &\approx \max_Q p(X \mid Q, W)P(Q \mid W) \\ &= \pi\left(q_{k_1}\right) \prod_{t=2}^{T} a_{q_{k\,t-1} q_{k_t}} \prod_{t=1}^{T} p\left(\boldsymbol{x}_t \mid q_{k\,t}\right) \end{aligned} \tag{2.8}$$

where $\hat{Q} = \langle q_{k_1}, \ldots, q_{k_t}, \ldots, q_{k_T} \rangle$ is the most probable state sequence obtained from the Viterbi

algorithm [Rabiner (1989)] for decoding and $\pi\left(q_{k_1}\right)$, $a_{q_{kt-1}q_{k_t}}$ and $p\left(\boldsymbol{x}_t \mid q_{kt}\right)$ have the usual meanings described in Section 2.1.2 in the context of HMM. The marginalization over all possible hidden state sequences is typically approximated just by using the most probable hidden sequence.

For a more detailed reading on HMM, conventional HMM-based ASR, and the neural network based hybrid connectionist approach to ASR, we refer the reader to the following resources: [Rabiner (1989); Jelinek (1998); Bourlard and Morgan (2012)].

### 2.1.4 Speech Units for Acoustic Modeling: Words, Phonemes, and Senones

Determining the degree of fineness (i.e., granularity) of the speech units for acoustic modeling plays a critical role for ASR systems. For example, given that there is enough data available for modeling each class of unit properly, speech units should be chosen in such a way that the probability distributions of distinct classes (over acoustic features) are distinct. In addition, the speech unit should be easily *generalizable* for the unseen new words in the test sentences.

#### Words

Modeling speech at the word level seems to be the most intuitive choice, considering they are language specific units having different semantic meanings. As there should be enough data available for modeling each word (i.e., each class of speech unit) properly, this unit selection requires a significant amount of training data with many (context-dependent) variations of each word. Therefore, learning robust and accurate word models with good generalization power on new unseen words in the test data is not scalable and not feasible for Large-Vocabulary Continuous Speech Recognition (LVCSR).

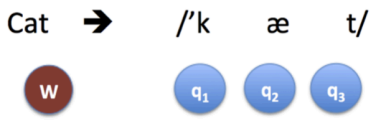



***Figure 2.2:*** *Example of the phoneme sequence for a word. Figure has been reproduced from [Dighe (2019)] with permission.*

#### Phonemes

Phonemes are linguistically distinct speech units without any semantic meaning. The phoneme units are defined only with respect to the constituent sound, and are therefore not language-dependent. That is, different languages might have different sets of phonemes while still sharing some common ones.

Modeling speech at phoneme level is more logical and applicable than word-level. There

are nearly 40 phonemes in English, compared to about 170,000 words. Hence, modeling each phoneme properly does not require huge amounts of training data and does not pose scalability issues for LVCSR. Actually, words can be modeled as sequences of phoneme models concatenated together (as shown in Figure 2.2). A linguist can prepare a dictionary in which all the words in the vocabulary are mapped to their phonetic sequences, allowing the phoneme-based model to generalize on the unseen words in the data.

A typical phonetic model is based on a 3-state left-to-right HMM topology (Figure 2.3) where each state can be modeled using a separate GMM. The states of the phone-HMM are sub-phonetic units which model the beginning, middle, and end of the phone. In addition, a similar HMM model is used to model the silence class.

In phone-based acoustic modeling, the acoustic features are typically modeled using distinct probability distributions belonging to the phone-states. A shortcoming of this monophone modeling approach is that we consider all the instances of a phoneme in the data to be acoustically similar, ignoring the impacts of the surrounding phoneme context and variations in the coarticulation mechanism during phoneme's acoustic realization. Therefore, the monophone modeling tries to fit all instances of the phoneme under a common probability distribution which results in possibly poor and inaccurate acoustic modeling. Conventionally, modeling of phones is addressed by context-dependent sub-phonetic modeling described below.

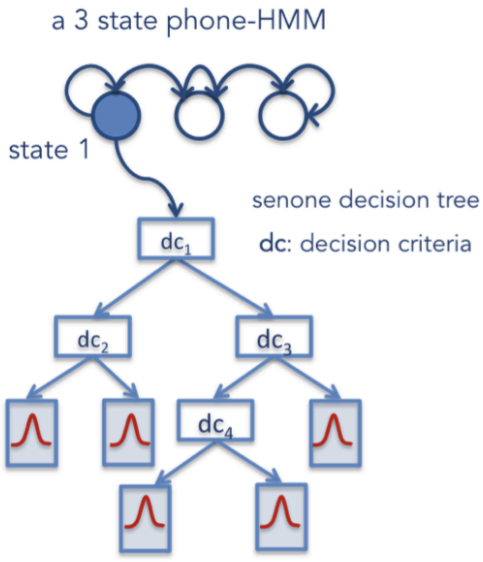



***Figure 2.3:*** *Example of a traditional 3-state triphone HMM and the senone decision tree. Figure has been reproduced from [Dighe (2019)] with permission.*

#### Senones

To incorporate context-dependency in phoneme modeling, monophone HMMs are replaced with triphone HMMs. A triphone has a unique left and right context phoneme around the central phoneme. States of the triphone HMMs are clustered across different phonetic models if the phonetic context is similar. This clustering (i.e., state tying) limits the number of different models to be trained.

Acoustic data assigned to each triphone state is further split using a decision tree [Young et al. (1994); Hwang et al. (1996)] such that each node asks a linguistic question to reduce the entropy or increase the likelihood of the data after split. The leaves of the decision tree are referred to as *senones*. A senone decision tree is illustrated in Figure 2.3. Senones are the most commonly used units for acoustic modeling in LVCSR systems because they provide significant improvement in ASR performance over the monophone acoustic models.

### 2.1.5 DNN-HMM Hybrid Acoustic Models

In a hybrid DNN-HMM ASR system [Bourlard and Morgan (2012)], DNN takes contextual (i.e., context-appended) acoustic feature vector as input and predicts the posterior probabilities of all senone classes at the output layer; thus, the frame likelihood required in (2.8) has to be indirectly estimated as follows:

$$p\left(\mathbf{x} \mid q_k\right) \sim \frac{p\left(\mathbf{x} \mid q_k\right)}{p(\mathbf{x})} = \frac{P\left(q_k \mid \mathbf{x}\right)}{P\left(q_k\right)} \tag{2.9}$$

The state posterior probability $P\left(q_k \mid \mathbf{x}\right)$ is at the output of the DNN acoustic model and $P\left(q_k\right)$ is the prior probability of the state $q_k$ computed from its frequency count in the training data, yielding to an estimate of the scaled likelihood $\frac{p(\mathbf{x}|q_k)}{p(\mathbf{x})}$.

Thanks to DNN, through multiple layers of non-linear transformations, the mapping from the acoustic features to the state posterior probabilities is done. DNN acoustic model can be a feedforward neural network or any other architecture like convolutional neural network (CNN) [Golik et al. (2015)], time-delay neural network (TDNN) [Peddinti et al. (2015)] or recurrent neural network (RNN) [Sak et al. (2014)].

DNN-HMM hybrid systems cannot be trained from scratch, without pre-existing frame-level alignments. Typically, this is handled by an off-the-shelf GMM-HMM system which force-aligns the senone sequences over the training utterances using their ground-truth text transcript under Viterbi algorithm. DNNs are trained using the error backpropagation algorithm [Rumelhart et al. (1985)] which utilizes the chain rule to compute the derivative of the loss with respect to each trainable parameter and updates them accordingly to the learning objective.

DNN acoustic models can be trained either for minimizing the frame-level (i.e., frame-wise) senone classification error, or for minimizing the sequence (i.e., sentence)-level error. Frame-level training is usually done by minimizing a Cross Entropy (CE) loss function, whereas sequence-level training is done with sequence-discriminative loss functions such as state-level Minimum Bayes Risk (sMBR) [Veselỳ et al. (2013)] and Maximum Mutual Information (MMI) [Povey et al. (2008)]. Here, we provide details for CE and MMI objectives only, as they are related to the work presented in this thesis.

**Cross Entropy Loss:** On a training example, if the target posterior vector is **t** and the DNN predicts a posterior vector **o**, then the CE loss is :

$$\mathscr{L}_{CE} = -\sum_{k}^{K} t_k \log(o)_k \tag{2.10}$$

where $t_k$ and $o_k$ are the $k^{th}$ components of DNN target and output vectors, respectively. By minimizing the CE loss over the whole training examples, we minimize the Kullback-Leibler (KL) distance between the target probability distribution and the DNN acoustic model output distribution.

However, DNN acoustic models trained using CE criterion are suboptimal (in terms of word recognition performance), as speech recognition is inherently a sequence classification problem. Sequence-level (i.e., sequence-discriminative) handles this issue, obtaining significant performance improvements [Veselỳ et al. (2013)]. Traditionally, this is performed by retraining a CE-trained model using one of the sequence-discriminative criteria (MMI, in our case).

**Maximum-Mutual Information Criteria:** As its name suggests, the aim is to maximize the mutual information between the acoustic observation $X$ and the word sequence $W$

$$\begin{aligned} \mathscr{L}_{MMI} &= \log \frac{p(X,W)}{p(X)P(W)} \\ &= \log \frac{p(X \mid W)P(W)}{\sum_{\hat{W}} p(X \mid \hat{W})P(\hat{W})} - \log P(W) \end{aligned} \tag{2.11}$$

If the observation $X$ and the word sequence $W$ are completely independent according to the model, the equation is equal to 0 implying that $X$ and $W$ are unrelated. Assuming $P(W)$ is independent of the model parameters $\theta$, (2.11) can be simplified as:

$$\mathscr{L}_{MMI} = \log \frac{p(X \mid W,\theta)P(W)}{\sum_{\hat{W}} p(X \mid \hat{W},\theta)P(\hat{W})} \tag{2.12}$$

Intuitively, MMI maximizes the probability of the ground-truth transcription, while minimizing the probability of all other transcriptions (hence, it is a sequence-discriminative objective).

Theoretically, the denominator sum should be computed over all possible word sequences. However, to reduce the computation cost in Kaldi [Povey et al. (2011)], this sum is constrained by decoding the lattice which contains the most probable hypothesis. Also, MMI training in Kaldi is traditionally performed by re-training a CE-based acoustic model which has been trained as the seed model to generate the decoded lattice. Thus, the denominator can be approximated as:

$$\begin{aligned} \sum_{\hat{W}} p(X \mid \hat{W}, \theta) P(\hat{W}) &= \sum_{\hat{W}} p\left(X \mid \mathbb{M}_{\hat{W}}, \theta\right) \\ &\approx p\left(X \mid \mathbb{M}_{den}, \theta\right) \end{aligned} \tag{2.13}$$

where $\mathbb{M}_{den}$ (denominator graph) is, due to Kaldi convention, is an HMM-based graph that includes all possible word sequences in the decoded lattices.

## 2.2 Lattice-Free MMI based ASR

Lattice-Free Maximum Mutual Information (LF-MMI) [Povey et al. (2016)] approach allows to perform sequence discriminative training from scratch, without the CE-based pre-training step. LF-MMI system has several computational advantages over traditional sequence discriminative training approaches (e.g., sMBR, MMI) and it provides better recognition performance.

In this thesis, we deploy the state-of-the-art LF-MMI acoustic models as *baseline systems* (Section 2.5). As these baseline systems mostly consist of Time-Delay Neural Network (TDNN) based building blocks, we here elaborate the architectural properties with special emphasis on TDNN models. The blueprint for the baseline LF-MMI acoustic model is illustrated in Figure 2.4.

Similar to a convolutional network, the TDNN models temporal dependency in the time domain. Compared to a recurrent network, TDNN can be parallelized easier and requires less training time than feed-forward DNN. The input of each unit in TDNN layers is expanded out spatially in few sequential units from the previous layer. Consequently, lower layers gain an understanding of a narrow context while higher layers gain an understanding of a broader temporal context.

To compress the network layers, a *factored* form of TDNN derived from singular value decomposition (TDNN-F) is introduced as an improvement over TDNN. TDNN-F block comprises a linear-affine sequence of operations that act like a bottleneck transforming the input vector into an intermediary vector and then back into the output vector. In addition, each TDNN-F block ends with a summation operation that adds the output of the current processing block to the down-scaled output of the previous TDNN-F block. In this sense, the summation operation resembles residual connections.

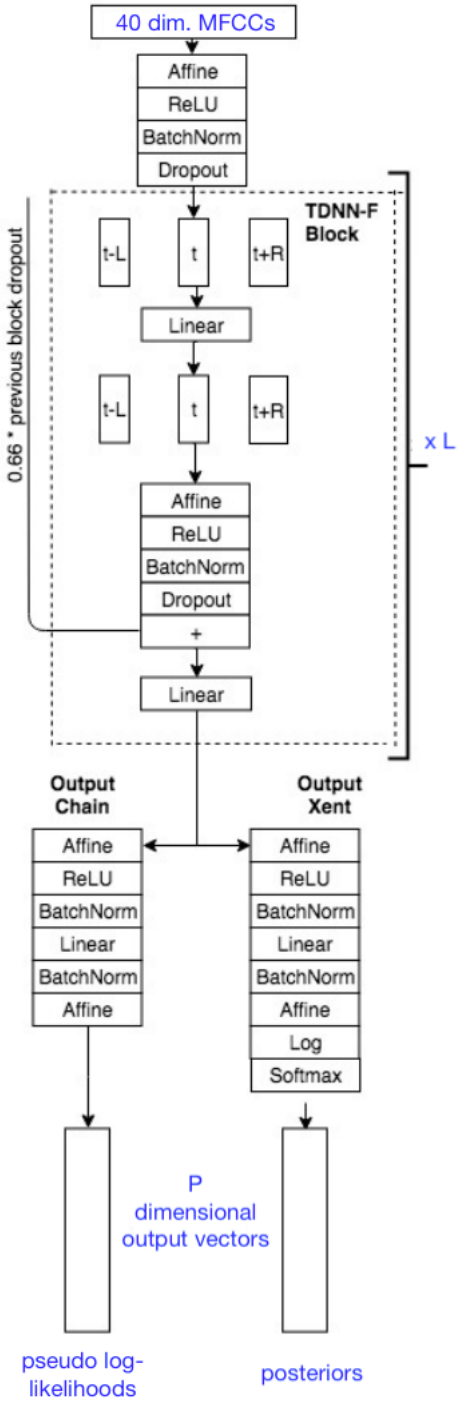



***Figure 2.4:*** *The blueprint for the model configuration of the baseline LF-MMI acoustic model is illustrated. While experimenting with different datasets (Section 2.3), except for small differences such as the number of TDNN-F blocks (L), output vector dimensionality (P) etc., we stick to the illustrated model configuration. Figure has been adapted from [Georgescu et al. (2019)].*

Except for the first TDNN-F block, which processes only the input from the current time-frame, all other TDNN-F blocks perform 1-D temporal convolution, processing input vectors at times t-$L$, t, and t+$R$. The input vectors at times t-$L$ and t are spliced together into the linear layer, while the input vectors at times t and t+$R$ are spliced together into the affine layer.

The LF-MMI models have two output blocks: the first branch is chain based (i.e., Output Chain in Figure 2.4) and the second branch uses the CE criteria (i.e., Output Xent in Figure 2.4). Each output branch is composed of affine, ReLU, batch normalization and affine layer again. The primary difference between these two blocks comes from the presence (or absence) of the final log-softmax operation. The output without softmax is used in inference, although both blocks are used in the training process.

The output features of the LF-MMI system (i.e., the output from Output Chain) are called *pseudo-log-likelihoods*. We will refer to them as *LF-MMI log-likelihoods* in the rest of the thesis. The dimension of the output features (i.e., $P$ in Figure 2.4) is equal to the number of *senones* (Section 2.1.4), which is determined by Kaldi speech processing toolkit [Povey et al. (2011)].

## 2.3 Datasets

Details of all databases used in this thesis are given below.

The experiments based on low-rank and sparse modeling of LF-MMI senone log-likelihoods in Chapter 4 and of MFCC acoustic features in Chapter 5 are conducted on AMI Meeting corpus with parallely recorded clean and far-field conversational speech recordings.

The experiments for transferring the acoustic space knowledge from AMI Meeting corpus (pre-learned in Chapter 6) to pathological speech in Chapter 7 are performed on UA-Speech dataset.

### 2.3.1 AMI

The AMI corpus[1] contains recordings of spontaneous conversations between a group of participants in meeting scenarios. The meeting scenarios have been designed such that the participants freely discuss and debate over some ideas. The meetings were recorded in English, although the speakers were mostly non-native. AMI corpus provides audio recordings from close-talk as well as far-field microphones. Due to the conversational style of speaking and the speakers frequently overlapping and interrupting other speakers' speech, the AMI corpus has proved to be a challenging task in recent large vocabulary ASR research.

In this thesis, we focus on speech data recorded using close-talk and far-field microphones. The close-talk microphone speech is termed as individual headset microphone (IHM) condition in AMI, whereas the far-field microphone speech is termed as the single distant micro-

[1] http://groups.inf.ed.ac.uk/ami/corpus/

phone (SDM) condition. All meeting rooms had eight far-field microphones in a circular array between the meeting participants. The first microphone (mic-id 1) is typically used as the source of SDM data for far-field ASR. Both the close-talk and far-field speech streams have been recorded parallelly. They are time synchronized, and the word transcripts are obtained by force-aligning using a speech recognition system.

#### Data protocol

The dataset is available at 16kHz sampling rate with nearly 100 hours of meeting recordings divided approximately as 81 hours train set, 9 hours development and 9 hours evaluation set. We use the development set for tuning the hyper-parameters of our proposed models.

***Table 2.1:*** *Details of AMI database. The number of utterances (and duration in hours) in AMI training, development and test set.*

| Detail | Count (partition-wise) | | |
|---|---|---|---|
| | Train | Dev | Test |
| Speech data (in hours, approx.) | 81 | 9 | 9 |
| Number of utterances | 108,221 | 13,059 | 12,612 |
| Running words | 802,604 | 94,914 | 89,635 |

### 2.3.2 UA-Speech

The UA-Spech [2] database is intended to promote the development of user interface for speakers with neuromotor disorders and spastic dysarthria.

The database includes 15 speakers with cerebral palsy (denoted as DYS) and 13 age-matched healthy control speakers (denoted as CTL). Dysarthric speakers were selected based on self-report of either speech pathology or cerebral palsy. However, before the data were included in the dataset the diagnosis of dysarthria was informally confirmed by a certified speech-language pathologist through audio recordings.

Speakers read isolated words including digits (10 words with 3 repetitions), letters (26 letters of International Radio Alphabet with 3 repetitions), computer commands (19 word processing commands with 3 repetitions), common words (the most common 100 words in the Brown corpus [Francis and Kucera (1979)] with 3 repetitions), uncommon words (300 words selected from Project Gutenberg novels using an algorithm that sought to maximize biphone diversity (e.g., naturalization, faithfulness, frugality etc.) with 1 repetition).

[2] http://www.isle.illinois.edu/sst/data/UASpeech/

#### Data protocol

The database is available at 16 kHz sampling rate with 70 hours of recordings of the isolated words (i.e., digits, letters, computer commands, common words, uncommon words). The recordings were made with 7 microphones from 15 dysarthric speakers (about 40 hours in total) and 13 age-matched healthy (i.e., without any speech impairment) control speakers (about 30 hours in total).

In our experiments, we used the re-segmented version of the corpus from [Xiong et al. (2019)] where excessive portions of silence if divided into three distinct blocks, two of which (Block 1 and 3 in Table 2.2) were commonly used as the training set, while models were evaluated on Block 2. Each block consists of 10 digits, 26 letters, 19 computer commands, 100 common words and 100 distinct uncommon words.

***Table 2.2:*** *Details of UA-Speech database. The number of utterances (and duration in hours) in UA-Speech training and test set with and without re-segment.*

| Sets(Spk) | Re-segment | Block 1 & 3 | Block 2 |
|---|---|---|---|
| CTL | | 46410(22.7 h) | 23205(11.1 h) |
| #13 | ✓ | 46403(19.8 h) | 23205(9.7 h) |
| DYS | | 49204(44.3 h) | 24731(21.7 h) |
| #15 | ✓ | 49204(27.3 h) | 24727(13.4 h) |

## 2.4 Evaluation Metrics

Details of the evaluation metrics used in this thesis are given below.

### 2.4.1 Word Error Rate

In this thesis, we mainly used Word Error Rate (WER) which is the most commonly used metric for measuring the performance of an automatic speech recognition system. Given the reference word sequences over some test data and the word sequences hypothesized by an ASR system, the word error rate is defined as:

$$\text{WER ( in \%)} = 100 \times \frac{\text{Substitutions + Deletions + Insertions}}{\text{Number of words in the reference}} \tag{2.14}$$

where the numerator has a count of substitutions, deletions and insertions in the hypothesized word sequences as compared to the reference sequences. WER is typically presented as a percentage, and a lower WER signifies better performance in speech recognition (i.e., the lower, the better).

### 2.4.2 Frame-level Phone Accuracy

In addition to WER, in this thesis, we used the frame-level phone accuracy for evaluating the performance of various features (e.g., acoustic features, different encodings from autoencoders) in terms of their discriminative power when used for training a simple 2-layered fully-connected classifier. Higher frame-level phone accuracy signifies better performance in frame-level phone classification task (i.e., the higher, the better).

This accuracy is defined as follows:

$$\text{Frame-level Phone Accuracy ( in \%)} = 100 \times \frac{\text{Number of correctly predicted phone frames}}{\text{Total number of frames}} \tag{2.15}$$

## 2.5 Baseline Systems

In this thesis, while experimenting with different datasets (Section 2.3), we mainly sticked to the LF-MMI acoustic model configuration illustrated in Figure 2.4, except for small differences such as the number of TDNN-F blocks (L), output vector dimensionality (P) etc.

All baseline LF-MMI acoustic models take 40 dimensional high-resolution MFCCs (extracted from frames of 25 ms length and 10 ms shift) as input.

For AMI Meeting corpus (Section 2.3.1), baseline LF-MMI acoustic models contain 15 TDNN-F blocks (i.e., $L = 15$ in Figure 2.4) with dimensionality of 2136 and bottleneck dimension of 210. In addition, the linear component dimensionality is 512. The number of senones (i.e., P in Figure 2.4) is 176 for both close-field (IHM) and far-field (SDM) portions of the data. The baseline WERs are 19.2% and 41.1% for IHM and SDM, respectively.

For UA-Speech database (Section 2.3.2), baseline LF-MMI acoustic model contains 13 TDNN-F blocks (i.e., $L = 13$ in Figure 2.4) with dimensionality of 1024 and bottleneck dimension of 128. In addition, the linear component dimensionality is 192. The number of senones (i.e., $P$ in Figure 2.4) is 1696 and 1728 for healthy (CTL) and pathological (DYS) speech portions of UA-Speech database. The baseline WERs are 18.5% and 38.9% for CTL and DYS, respectively.

## 2.6 Conclusion

In this chapter, we provided a brief background on automatic speech recognition (ASR), presenting the key components in the ASR pipeline with special emphasis on acoustic modeling. We introduced the LF-MMI recognition system (Figure 2.4) which stands for the state-of-the-art acoustic model in the thesis. Finally, we gave details of the databases, evaluation metrics and the configuration for the LF-MMI baseline systems.

In the thesis, we work with different sparse autoencoder models (elaborated in Chapter 3) for learning informative and interpretable speech representations, mainly towards improving ASR in acoustically mismatched conditions. In essence, autoencoder is inserted in the ASR pipeline (Figure 2.1) as an additional component after the LF-MMI acoustic model as in Chapter 4 or before the LF-MMI acoustic model component as in Chapter 5. In Chapter 6, we further explore different sparse autoencoders for the representation learning problem to improve the robust speech modeling. Finally, in Chapter 7, we specifically focus on plugging the pre-trained, off-the-shelf sparse autoencoder (Chapter 6) in a different ASR pipeline which is originally designed for pathological speech recognition.

# 3 Background on Autoencoders

In this chapter [1], we provide background information about autoencoders with a particular focus on the sparse autoencoders. First, we briefly remind the reader the relation of autoencoders with Principal Component Analysis (PCA). Then, we highlight the importance and the consequences of design choices (e.g., increasing depth or width, employing different regularization constraints) in autoencoders. Finally, in the last section, we look into different sparse autoencoder configurations and viewpoints on sparsity.

## 3.1 Autoencoders Reloaded

Autoencoders (AEs), previously called "auto-associative multilayer perceptrons", are neural networks whose goal is to reconstruct $d$-dimensional input (i.e., observation) vectors as $d$-dimensional output vectors. AE consists of two main parts: encoder and decoder. When $d$-dimensional input is passed through the encoder, code (i.e., encoding, embedding) is extracted on the latent encoding layer. This code is then fed to the decoder to create the reconstruction for the original input at the output layer of the autoencoder. In general, AE's objective is to minimize the Mean Square Error (MSE) between the original input and its reconstruction.

Depending on the model configuration, autoencoders are named *undercomplete*, if the encoding layer has lower dimensionality than the input, *overcomplete*, if the encoding layer has the same or more units than the input, *shallow*, if there exists only one hidden layer, and *deep*, if there exists more than one hidden layer in the model configuration.

The availability of large amounts of data across multiple disciplines associated with significantly increased memory and computation resources has renewed the interest in Deep Neural Networks (DNNs). In this context, autoencoders, which were initially utilized for linear bottleneck feature extraction, are today also widely explored as a potentially strong nonlinear

[1] This chapter is partially based on the following publication:
Bourlard, H., and Kabil, S. H. (2022). Autoencoders reloaded. Biological Cybernetics, 116(4), 389-406.

feature extraction model. This enables notably successful use of unsupervised deep learning related to the autoencoders, including word embeddings [Mikolov et al. (2013)], a method used to represent discrete variables as continuous vectors, and linear/nonlinear transformers, mostly in use for speech and natural language field [Wolf et al. (2020)]. Consequently, DNN applications have expanded from image processing to speech recognition, natural language processing, time-series forecasting and neuroscience studies.

Our objective for this thesis is to devise sparse architectures in an unsupervised manner for learning meaningful representations of speech which can also be utilized for different speech tasks (e.g., speech recognition, pathological speech recognition and modeling). We address this feature learning (i.e., representation learning) problem by means of autoencoders, as we argue that they are the simplest (and most intuitive) neural network architectures for unsupervised learning. Along this line, we first briefly present shallow undercomplete autoencoders, then, we highlight the importance of keeping the modeling capacity of the autoencoder under control with different regularization instruments, among which we specifically focus on sparsity constraint. Lastly, we review sparse autoencoders and the concept of sparsity for learning meaningful representations.

### 3.1.1 Shallow Undercomplete Autoencoders

*Shallow undercomplete autoencoder* is an autoencoder with only one fully-connected hidden layer which has lower dimensionality than the input. In [Bourlard and Kamp (1988)], it was shown that a shallow undercomplete autoencoder, with linear decoder and MSE loss function, learns the weights that span the same subspace as the one spanned by the principal component vectors.

[Bourlard and Kamp (1988)] demonstrated that an autoencoder trained (with the usual error back-propagation (EBP) algorithm [Rumelhart et al. (1985)]) to minimize MSE is equivalent to linear PCA [Hotelling (1933); Jolliffe (1986)], or alternatively and equivalently, Singular Value Decomposition (SVD) [Golub and Reinsch (1971)] of the input data matrix, even with nonlinearities in the hidden layer. This has often been subjected to misperceptions, or argumentations from the neural network community. Yet, [Bourlard and Kabil (2022)] further validated this conclusion by experimenting on larger datasets which were not available at that time.

Hence, in the context of shallow undercomplete autoencoder topology (Figure 3.1), when MSE (3.1) is used as loss function, PCA yields the optimal solution. However, it is likely that non-optimal MSE solutions, corresponding to local minima or to constrained solutions, result in higher-quality "features" (i.e., hidden layer activations, encodings, embeddings) as MSE is not necessarily related to classification or regression performance (i.e., performance on a downstream task).

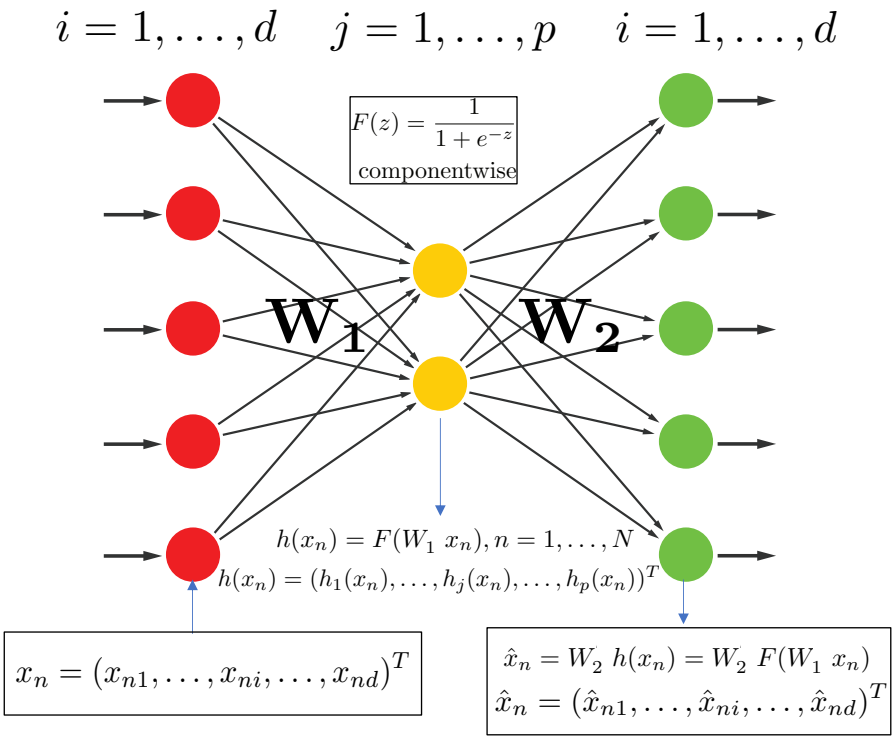



***Figure 3.1:*** *Shallow undercomplete autoencoder learns to span the same subspace as PCA under certain conditions such as linear decoder (i.e., linear output layer), MSE as loss function and real-valued input data. In addition, having a smaller code dimension than the input dimension forces the autoencoder to learn the salient features as encodings from the training data.*

### 3.1.2 Going Deep or Going Overcomplete

Shallow undercomplete autoencoder learns to span the same subspace as PCA under certain conditions such as linear decoder (i.e., linear output layer), MSE as loss function and real-valued input data. Having smaller code dimension than the input dimension (i.e., being undercomplete) forces the autoencoder to learn the salient features as encodings from the training data. Apart from this, of course, there exist other ways to design autoencoders.

While building neural networks, the key design decisions are usually choosing the depth of the network and the width of each layer. Deeper networks are often able to use far fewer units per layer and generally far fewer parameters; hence, frequently generalizing to the test data. In addition, depth can exponentially reduce the computational cost of representing some functions [Goodfellow et al. (2016)].

Apart from the shallow autoencoders (illustrated in Figure 3.1 and Figure 3.2), autoencoders can also be deepened, consisting of several hidden layers in the encoder and decoder. However, deep architectures can also tend to be harder to optimize [Bengio et al. (2007); Goodfellow et al. (2016)]. One of the earliest strategies to train a deep autoencoder is to greedily pretrain the

deep architecture by training a stack of shallow autoencoders. So, even when the end goal is to train a deep autoencoder, shallow autoencoders remain useful [Bengio et al. (2007)]. Other aspects to keep in mind while building an autoencoder are the use of nonlinear activations in the encoder and decoder, the composition of different unit types (e.g., convolution layers, recurrent layers), the higher dimensional encoding layer (i.e., being overcomplete) and the depth of the encoder and decoder.

However, it is important to keep in mind that if the autoencoder is given too much modeling power, it can simply learn the identity function (i.e., perfectly reconstructing the input data at the output layer) between the input data and output reconstruction. This phenomenon is also known as identity mapping. In this case, some form of constraint (e.g., sparsity, contraction, noise) should be introduced during training to guide the autoencoder for learning meaningful encodings.

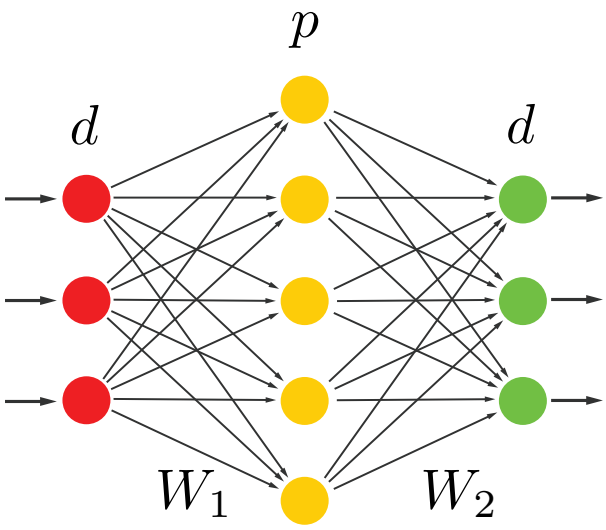



***Figure 3.2:*** *Shallow overcomplete autoencoder can easily overfit and do perfect reconstruction (i.e., identity mapping) while learning meaningless encodings on its hidden layer. To avoid this phenomenon, different regularization methods can be applied on the model.*

### 3.1.3 Regularized Autoencoders

Autoencoders with constraints to restrict the modeling capacity are known as *regularized autoencoders*. Modeling capacity denotes the complexity of the relationships in the data (i.e., patterns) that the model can express. The most common way to estimate the capacity of a model is to count the number of parameters, as more parameters indicate higher capacity. Hence, overcompleteness increases the modeling capacity of the autoencoder.

In other words, given the input data, if the autoencoder tends to overfit, it can simply end up learning the identity function (i.e., reconstructing the input data at the output layer perfectly) which is simply uninteresting. Thus, high modeling capacity is not always desired. To avoid

this trivial identity function, different forms of constraints are exploited.

These constraints are usually in the form of additional regularizer term(s) in the AE loss function. More precisely, when the loss function for the autoencoder training (3.1) is written as follows, with $\hat{x}_n$ and $x_n$ being reconstructed and original data instance, respectively,

$$L_{\text{AE}} = \frac{1}{N} \sum_{n=1}^{N} (x_n - \hat{x}_n)^2 \tag{3.1}$$

Then, the loss function for the regularized autoencoders can be merely stated as

$$L_{\text{RAE}} = \frac{1}{N} \sum_{n=1}^{N} (x_n - \hat{x}_n)^2 + \lambda L_{\text{reg}} \tag{3.2}$$

where $\lambda$ denotes the regularization weighting term, tuning the penalty term $L_{\text{reg}}$ with respect to the reconstruction loss (3.1).

#### Contractive autoencoders

Contractive Autoencoder (CAE) [Rifai et al. (2011)] uses an additional penalty term in the loss function (3.3). The penalty is the squared Frobenius norm of the Jacobian matrix of the encoder. In other words, the penalty term is the sum of the squares of all first-order partial derivatives for all input vectors.

$$L_{\text{CAE}} = L_{\text{AE}} + \lambda \sum_{n=1}^{N} \left\| J_f(x_n) \right\|_F^2 \tag{3.3}$$

This Jacobian term promotes local invariance to displacements and alterations in many directions around the training samples (i.e., training instances, training examples) so that the model gets less sensitive to the small perturbations in the input data. The penalty term forces the autoencoder to extract encodings whose all dimensions are contracted. However, at the same time, the reconstruction error prevents the model from contracting the dimensions along the (true) data manifold.

#### Denoising autoencoders

Denoising Autoencoder (DAE) [Vincent et al. (2008)] uses stochastic corruption of the (clean) input data $x_n$ as regularizer during model training. The corruption of the input, resulting in $\tilde{x}_n$, can be achieved through additive isotropic Gaussian noise, salt and pepper noise for gray-scale images, and/or masking noise (i.e., setting some randomly chosen inputs to 0 independently per instance).

Denoising autoencoder adopts a different criterion to evaluate the performance of the reconstruction (3.4), where $x_n$ and $g(\tilde{x}_n)$ denote the original clean data instance and reconstruction

for the associated corrupted data instance such that $g(\tilde{x}_n) = F(W_2 h(\tilde{x}_n))$, respectively.

$$L_{\text{DAE}} = \frac{1}{N} \sum_{n=1}^{N} \left( x_n - g\left(\tilde{x}_n\right) \right)^2 \tag{3.4}$$

In training, the model takes partially corrupted instances as input and tries to reconstruct the original, clean instances. This forces the model to identify the true data manifold.

### Variational Autoencoders

Variational Autoencoder (VAE) [Kingma and Welling (2019)] replaces the deterministic functions in the autoencoder configuration by stochastic mappings. The encoder does not map each instance to a single point in the latent embedding space, but to a distribution instead, which is usually a normal distribution, defined by its mean and standard deviation. Then, reconstruction is produced by sampling from that distribution and propagating the results through the decoder network. Since VAE allows sampling from the learned distribution, its applications usually involve generating new data instances [Dosovitskiy and Brox (2016)].

VAE assumes that a latent, unobserved random variable $z$ exists, which by some random process leads to the observation $x$. Its objective is thus to approximate the distribution of the latent variable given the observations. The loss function of VAE can be decomposed into terms of single data points. For the sake of simplicity, in ( 3.5), the loss function is for one datapoint $x_n$.

$$\begin{aligned} &\mathscr{L}_{\text{VAE}}(\theta, \phi; \mathbf{x_n}) = \\ &\quad \text{KL}\left(q_\phi(\mathbf{z} \mid \mathbf{x_n}) \| p_\theta(\mathbf{z})\right) - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x_n})}\left[\log p_\theta(\mathbf{x_n} \mid \mathbf{z})\right] \end{aligned} \tag{3.5}$$

where $q$ is the distribution approximating the true latent distribution of $z$, and $\theta$, $\phi$ are the parameters of encoder and decoder distributions, respectively.

In (3.5), the first term acts as a regulariser between the encoder's distribution and the standard normal distribution. That is, if the latent representations $z$ produced by the encoder are different from those from normal distribution $p_\theta(\mathbf{x})$, this term penalizes the loss. Whereas, the second term in ( 3.5) is the reconstruction loss, promoting the decoder to reconstruct the data $x_n$. If the reconstruction does not comply with the input data well, it can be said that the decoder parameterizes a distribution which fails to model the true distribution of the data.

To sum up, different constraints (e.g., sparsity, contraction, noise) restrict the modeling capacity, and hence guide the autoencoders (e.g., contractive AE, denoising AE) for learning meaningful encodings. Sparse autoencoders are more prominent to exploit the intrinsic properties of speech data (Section 1.1); hence, in the next section, we specifically focus on sparsity as primary constraint, and the use of sparse autoencoders for learning meaningful representation.

## 3.2 Sparse Autoencoders

Different models have been referred to as sparse autoencoders (sparse AE), as there is a lack of consensus in literature regarding the definition. Thus, in this section, we describe some of the well-known autoencoders for learning sparse representations. In autoencoders, sparsity is typically enforced by sparsifying the weights [Gupta and Majumdar (2016); Wan et al. (2013)] and/or sparsifying the hidden unit activations (i.e., encodings). Here, we mainly categorize the sparse AEs based on the explicit and implicit regularization features present in the model.

For the case of *explicitly constrained sparse AE*, penalty term is added to the loss function. A simple sparse AE can, for example, be intuitively built using $\ell_1$ norm regularization on the hidden unit activations (3.6), motivated by the sparse coding and dictionary learning [Olshausen and Field (1997)].

$$L_{\text{SAE}} = L_{\text{AE}} + \lambda \sum_{n=1}^{N} \|h(x_n)\|_1 \tag{3.6}$$

The goal of the dictionary learning is to find a dictionary $D \in R^{d\times p} : D = [d_1, \ldots, d_P]$ which is constrained with $\|d_i\|_2 \leq 1, \forall i = 1, \ldots, p$ and a representation $H \in R^{n\times p} : H = [h_1, \ldots, h_N]$, given the input data $X = [x_1, \ldots, x_N]$, $x_n \in \mathbb{R}^d$, such that all $\|x_n - Dh_n\|_2^2$ are minimized and all representations $h_n$, where $h_n = D^T x_n = h(x_n)$, are zero in most places. This can be formulated as the following optimization problem:

$$\min_{D,h_n} \sum_{n=1}^{N} \|x_n - Dh_n\|_2^2 + \lambda \|h_n\|_0 \tag{3.7}$$

Analogous to Figure 3.2, in dictionary learning setup, dictionary $D$ stands for the decoder weights $W_2$, while the encoder and decoder weights are tied $W_1 = D^T$. Since (3.7) is NP-hard [Candès and Wakin (2008)], the $\ell_0$-norm $\|h_n\|_0$ is relaxed to the $\ell_1$-norm $\|h_n\|_1$. In addition, different variants of $\ell_1$ regularized sparse AEs are introduced in [Zhang et al. (2018)] whose aim is to smooth the regularization penalty.

Another commonly used regularization on the hidden unit activations is the Kullback-Leibler (KL) divergence formulated in (3.9). Sparse AE with KL divergence [Ng et al. (2011)] applies KL penalty to enforce sparsity on sigmoidal hidden unit activations (i.e., sigmoid function on the hidden layer). KL divergence [Kullback and Leibler (1951)] estimates the distance between predetermined desired average activation $\rho$ and the average activation for the hidden units. $p$ in (3.8) denotes the number of hidden units, in accordance with Figure 3.2.

$$L_{\text{SAE}} = L_{\text{AE}} + \lambda \sum_{j=1}^{p} \text{KL}\left(\rho \| \hat{\rho}_j\right) \tag{3.8}$$

$$\mathrm{KL}\left(\rho \| \hat{\rho}_j\right) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j} \tag{3.9}$$

The desired average activation $\rho$ is set to a very small value (e.g., 0.002). When $\hat{\rho}_j$ (i.e., average activation of hidden unit $j$ over training data) is close to $\rho$, KL penalty is expected to be low. However, in the opposite case, KL penalty grows rapidly, finally approaching to infinity when $\hat{\rho}_j$ gets close to 1.

For the case of *implicitly constrained sparse AE*, the sparsity constraint is not reflected in the loss function. Instead, it is maintained by altering the inner components of the model. k-sparse AE [Makhzani and Frey (2013)] is the most well-known example in this category. The model forces the hidden unit activations to be sparse by a pruning mechanism (i.e., top-k selection). In the encoding layer, $k$ hidden units with the highest activations are preserved while the activations from the rest of the hidden units are set to zero. This top-k selection introduces nonlinearity to the model, as it can also be seen as a variant of Rectified Linear Unit (ReLU) in which the threshold is the $k$-th largest activation (instead of zero). However, the performance is largely dependent on the hyperparameter $k$. It is observed that some of the hidden units may never be selected among top-k units, and thus never get updated during back propagation. This issue (i.e., dead atom problem in Section 3.2.3) is addressed with a scheduler for k during training. However, at inference time, $\alpha * k$ (where $\alpha > 1$, generally 3-4) is found to give the best result, indicating that the model is still not efficient at generalizing.

To rectify the aforementioned shortcomings in k-sparse AE, [Chen and Zaki (2017)] presents k-competitive AE. The model benefits from the concept of competitive learning [Rumelhart and Zipser (1985)] to boost the specialization of the hidden units. It uses hyperbolic tangent activation function which produces positive and negative activations in $[-1,1]$ range. $k/2$ units with the highest positive activation are selected as the positive winners (indicating high interest and specialization for the given input pattern) and $k/2$ units with the lowest negative activation are selected as the negative winners (indicating negative interest for the input pattern). Unlike k-sparse AE, the activations from the rest of the hidden units (i.e., losers) are summed, amplified by $\alpha$ and added onto positive and negative winners' activations, before being set to zero. In this way, the competition (and hence specialization) among hidden units is sharpened so that the active units (i.e., winners) are given chance to react more confidently to the input pattern while the inactive units are no longer in competition and can ensure not to react to the pattern in the future. During back propagation, the gradients first flow through the winner units in the hidden layer and then through the loser units thanks to $\alpha$-amplification connections. Hence, unlike k-sparse AE, the loser units are also given chance to receive feedback. In addition, it is stated that there is no need to tune $k$ in inference time, suggesting that k-competitive AE is better at generalizing, compared to k-sparse AE.

Sparsity can also be enforced on weights. Sparsifying weights is reported to help detect the meaningful relations while pruning the irrelevant ones. In sparsely connected autoencoders [Gupta and Majumdar (2016)], each unit is linked to only a limited number of units in

the following layer. It is worth mentioning that sparsely connected autoencoders learn the sparse connections deterministically, different from dropout [Srivastava et al. (2014)], and dropconnect [Wan et al. (2013)].

Furthermore, sparsity can be introduced via the choice of activation functions. For instance, ReLU function on hidden units, with its neuroscientific motivations, is intended to imitate the following properties of biological neurons [Glorot et al. (2011)]: (1) being completely inactive, for some inputs. (2) having output which is proportional to the input, for some inputs. (3) being inactive state most of the time (i.e., producing sparse activations).

In this thesis, we seek biologically motivated, simple yet elegant way of learning meaningful representations in an unsupervised and cost effective manner. For this reason, in the following subsections, we devote our attention to sparsity for its definition and benefits for representation learning, sparse overcomplete representations, sparse distributed representations with emphasis on population sparseness and life-time sparseness. And, finally we briefly touch on the sparsity measures used for analysis purposes in this thesis.

### 3.2.1 Sparsity

Definition of the term sparse may differ based on the perspective in applications. Intuitively, a sparse representation refers to the case where a relatively small number of coefficients carry a large proportion of energy (i.e., information). In mathematics, the terms “sparse” and “dense” usually denote the number of zero and non-zero elements in an array (e.g., vector or matrix). In the context of neural networks, hidden unit activations (i.e., encodings, representations) from a particular layer, the weights and the data can be described as sparse. For representation learning, from a computational perspective, sparse representations are beneficial for the following reasons [Glorot et al. (2011)]:

**Information disentangling**: Sparse representations are easier to manage for disentangling the factors explaining the variations in the data. On the other hand, a dense representation is highly entangled (i.e., intertwined, difficult to isolate the underlying components) because almost any change in the input intractably modifies most of the entries in the dense representation vector. For example, if a representation is both sparse and robust (to small changes in the data), the set of non-zero entries (i.e., features) is almost always distinct for the input. This brings in the aspect of interpretability which is useful for selecting accurate representations for different downstream tasks.

**Linear separability**: When the information is represented in a high-dimensional space, sparse representations are more likely to be easily separable with minimal computational effort. This point is further elaborated in Section 3.2.2.

**Efficient variable-size representation**: Sparsity allows the model to adjust the effective dimensionality of the representation for a given input. Different inputs may cover differing amounts of information and hence can be more effectively expressed using variable-size

representations. Thanks to sparsity, for a given input, only a subset of differing hidden units are active (i.e., differing entries in the representation vector are non-zero). In this way, the model can be conceptualized as consisting of an exponential number of linear models with shared parameters [Nair and Hinton (2010)].

**Distributed but sparse representation**: With respect to space efficiency, sparse representations present a good trade-off between dense distributed representations and local codes [Bengio (2009)]. More on this point is given in Section 3.2.3 with our motivation for reaching sparse distributed representations in this thesis.

### 3.2.2 Sparse Overcomplete Representations

Apart from sparsity, overcompleteness is another property that should carefully be examined. *Intuitively*, in the overcomplete scenario, the underlying latent components in the data are given a chance to *scatter* in the high(er)-dimensional representation space. Further, with the presence of sparsity, these components can be represented in a less aggregated manner. That is, the information is *distributed* among the entries of the sparse overcomplete vector and only a portion of its entries (i.e., dimensions) are critical and discriminative. Hence, when the information is represented in high-dimensional space, sparse representations are more easily separable and likely to be interpretable and informative.

Theoretically, [Kanerva (1988)] constitutes the pioneer study for understanding sparsity and overcompleteness together while proposing the *sparse distributed memory* concept for modeling the highly sparse representations in the brain. To clarify our standpoint, we mainly focus on two important properties from [Kanerva (1988)]: *orthogonalization* and *smooth latent space*. Orthogonalization is referred to when the data is projected onto a high(er) dimensional representation space, close data points in the original space are not necessarily mapped closely together. This is what was meant by "scattering" in our intuitive statement above.

However, during this projection, a learning function should also exist so that similar (i.e., potentially relevant) data points (sharing similar characteristics and latent components) can still be mapped relatively close. Hence, a data point and its (slightly) corrupted version are aimed to be mapped closely. The representation space with such behavior is referred to as *smooth latent space* in [Kanerva (1988)]. Smooth latent suggests that the latent components in the data are distributed in the representation space and hence can be traced (i.e., information disentangling). This is also referred to as "distribution of information" in our intuitive statement above. It is evident that there is a trade-off between orthogonalization and smooth latent spaces; thus, the balance should be maintained. Thus, sparse distributed memory can be viewed as realization of *locality-sensitive hashing*.

*Practically*, given a similarity measure (e.g., cosine similarity in the vector space) and a matching threshold, the impact of sparsity for robust matches among data points in the presence of noise is investigated in [Ahmad and Scheinkman (2019)]. Figure 3.3 illustrates the setup,

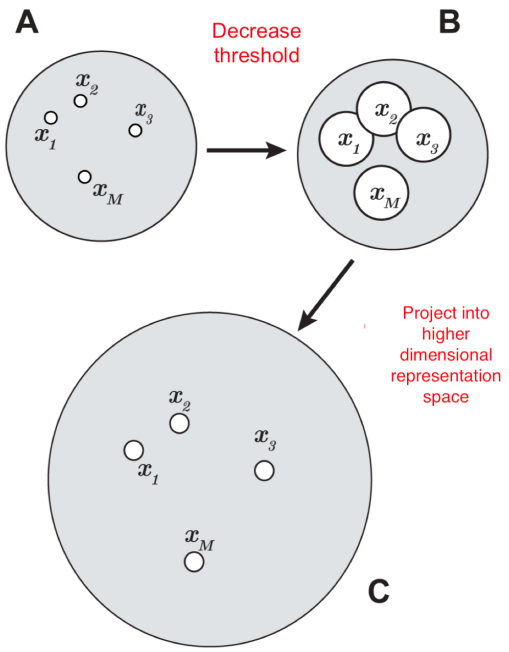



***Figure 3.3:*** *An illustration of the conceptual effect of decreasing the match threshold and increasing dimensionality. The large gray circles denote the universe of possible patterns. Each smaller circle represents the set of matches around one vector (data point). When threshold is high (A), very few random vectors can match with these vectors (hence, small white circles). As threshold is decreased (B), the set of potential matches increases (hence, larger white circles). Then, if dimensionality of the representation space is increased (C), the universe of possible patterns increases and the relative sizes of the white circles shrink rapidly. Figure adapted from Ahmad and Scheinkman (2019).*

the gray circles denote the universe of possible patterns to represent. Each smaller (white) circle represents the set of matches around one vector (i.e., data point). When the (matching) threshold is high (A in Figure 3.3), very few random vectors can match with the data vector (hence, small white circles). As the threshold is decreased (B in Figure 3.3), the set of potential matches expands (hence, larger white circles). Then, if dimensionality of the representation space is increased (C in Figure 3.3), the universe of possible patterns expands and the relative volume of the matching shrinks rapidly.

Accordingly, [Ahmad and Scheinkman (2019)] reports that *for a fixed sparsity level on distorted noisy environment, highly tolerant matches (i.e., with low(er) matching threshold) can be maintained without the cost of false positives (i.e., without matches with semantically irrelevant, random vectors).* Whereas, for the dense scenario, the false matches are reported to be relatively higher and unaffected by the dimensionality. This observation points out that both sparseness and overcompleteness (i.e., higher dimensionality) are necessary for robust matches and supports the insights from orthogonalization and smooth latent spaces.

### 3.2.3 Sparse Distributed Representations

After making our point on the sparse overcomplete representations, we clarify different viewpoints on sparsity. In [Ngiam et al. (2011)], these are reflected as follows:

**High dispersal:** In this scenario, no sparsity constraint is enforced, and dense representations are produced. The modeling capacity (referred to as universe of possible patterns in Figure 3.3) is very high and most of it is redundant. It is difficult to pinpoint which components are vital for understanding data characteristics (i.e., information disentangling is difficult.).

**Population sparseness:** Sparsity is enforced in the representation space. For each data point, the corresponding representation is indeed sparse. However, depending on the severity of the constraint, some entries of the representation vector (laying on the representation space) can always be zero. This phenomenon is known as the *dead atom problem* in sparse coding [Olshausen and Field (1996)]. Clearly, this behaviour is not desired.

**Life-time sparseness:** Life-time sparseness aims to prevent the aforementioned dead atom problem. Based on the data characteristics, only a small number of entries in the representation vector are desired to be active at any point in time. In other words, information is distributed over the dimensions of the representation space (i.e., entries of the representation vector) and each dimension covers some component (e.g., pattern) in the data, which indicates *specialization*. Hence, the overlaps of the contributing dimensions can reveal the conceptual similarities between different data points.

*It is worth mentioning that the population sparseness and life-time sparseness are not different types of sparsity but different viewpoints to sparsity.* Population sparseness focuses on the overall behavior of the unit population. It does not pay attention to a single unit's activation behavior in time. Hence, some dead units (e.g., dead atoms in sparse coding) may occur. Whereas, in the life-time sparseness, the same representation is inspected from a single unit point of view. The focus is on the activation behavior of each single unit during its life-time (e.g., training time for neural network) while still maintaining a sparse representation. In other words, the units are restricted not to be active or inactive all times (e.g., over all training data points), instead, they are forced to be more selective while making the decision to activate or not. Therefore, life-time sparseness is more successful for preventing dead atom problems and can more easily lead to Sparse Distributed Representations (SDRs), which are meaningful and discriminative by nature.

When learning meaningful intermediate representations from sparse autoencoders, we want these representations to be sparse, while still avoiding the dead atom problem. Therefore, we aim for learning sparse distributed representations (from speech data) by means of sparse autoencoders.

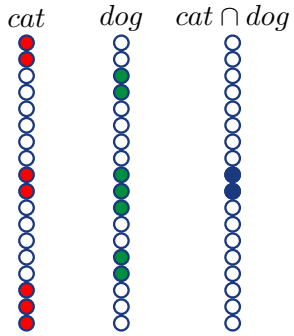



***Figure 3.4:*** *The illustration of sparse distributed representations for computing similarity relations among data points. Each red unit covers some underlying component which collectively stands for the concept "cat". The fact that overlaps exist and two of the units (shown in blue) are shared by "cat" and "dog" implicitly points to a particular similarity in terms of underlying components.*

In SDR, only a small number of units (i.e., neurons in neural network terminology, or dimensions in the representation space, or entries of the representation vector) from a much larger total field of units are active (i.e., contributing to the representation) at any point in time. Hence, the overlaps among active sets of units can point out the similarities between different data points. As illustrated in Figure 3.4, different data points are represented in the same representation space with only a few active (colored) units. For instance, each red unit covers some underlying component which collectively stands for the concept "cat".

The fact that overlaps exist and two of the units (shown in blue) are shared by "cat" and "dog" implicitly points to a particular similarity in terms of underlying components. This is useful not only for encoding unseen data points reasonably (i.e., out of distribution generalization), but also for the classifier to reach higher accuracy while exploiting the correlations, similarities between the representations in the supervised learning scenarios.

### 3.2.4 Population Sparseness and Life-time Sparseness

We aim for biologically motivated, simple sparse autoencoders for learning sparse distributed representations. Here, we reiterate over some of the different sparse autoencoders based on their viewpoint on sparsity (e.g., life-time sparseness, population sparseness).

As shown in Figure 3.5 illustrating the hidden unit activations over a batch of data points (i.e., data samples), $x$ axis acts as operation axis for population sparseness; whereas, $y$ axis acts as operation axis for life-time sparseness. This way, we can roughly say that life-time sparseness promotes temporal sparsity. Given that our research problem is based on speech, we put our imperative focus on maintaining life-time sparseness. However, we note that totally ignoring the population sparseness is not a good practice and in an ideal sparse autoencoder, these two viewpoints should be balanced.

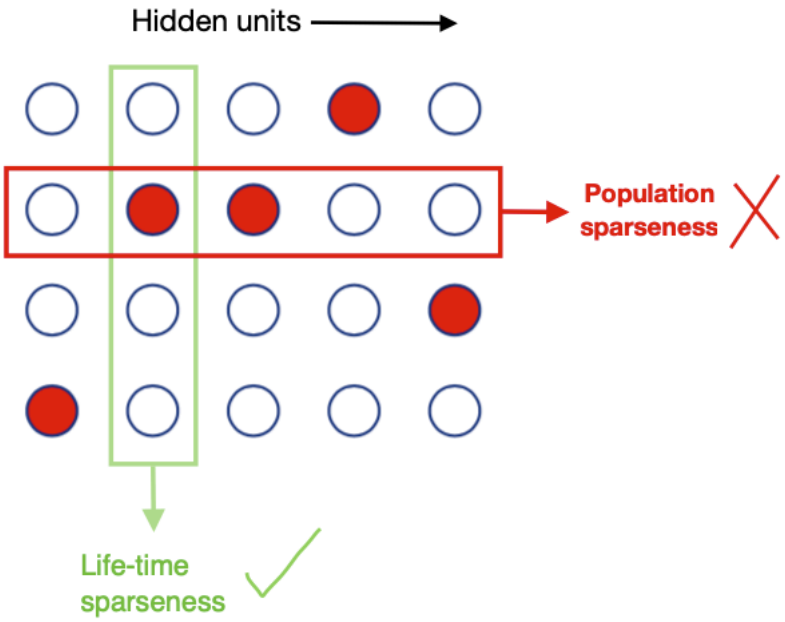



***Figure 3.5:*** *Illustration of hidden unit activation matrix in $R^{d \times p}$, where d and p stand for data batch size and number of hidden units, respectively. The red arrow represents the operation axis (i.e., x axis) for population sparseness. With population sparseness, some of the hidden units may not be active at all (i.e., dead unit problem) and/or some of the units may be active at all times (i.e., for all data samples). Hence, specialization of the hidden units is problematic. Whereas, with life-time sparseness (whose operation axis is denoted as green arrow), each hidden unit is desired to be active only a few times during its life-time (i.e., during training over data samples). Therefore, life-time sparseness does not lead to dead unit problem and promotes specialization. Given that our research problem is based on speech, we put out imperative focus on maintaining life-time sparseness.*

Sparse AE with $\ell_1$-norm regularization on the hidden unit activations mostly maintain population sparseness with tendency for extreme sparseness (i.e., dead units) and empirical effort to tune $\lambda$. This is mostly due to the nature of $\ell_1$-norm which pulls all the hidden unit activations' towards zero and is not interested in tracking or regularizing the individual hidden unit's activation(s) in time.

Sparse AE with KL penalty also fails to fully maintain life-time sparseness, as it operates on the average hidden unit activations, rather than individual unit activations in time. Minimizing the KL penalty leads the average of hidden unit activations (in time, over training samples) to be close to $\rho$. To satisfy this constraint, the activations from a single unit (e.g., j-th unit in (3.9)) may end up being constant and close to $\rho$ so that their average $\hat{\rho}_j$ is also close to $\rho$. This indicates that the model fails to boost the specialization of hidden units over the input data characteristics; hence, does not fully maintain life-time sparseness. In addition, KL penalty is only applicable to the autoencoders with sigmoid hidden unit activation.

It is desirable that each (hidden) unit is given equal chances to be active for learning robust sparse distributed representations. In the context of k-sparse autoencoders, active unit means unit being among top-k units. However, it also fails to maintain life-time sparseness with its delicate, empirically tuned k-scheduler mechanism. Meanwhile, k-competitive autoencoders have implicit mechanism based on the competitive learning insights, which can be taken as improvement upon k-sparse AE. However, in that case, the competition scheme to boost the specialization of hidden units and hence to maintain life-time sparseness is only applicable to the models with hyperbolic tangent activation.

### 3.2.5 Sparseness Measures

Even though it was first studied in computational neuroscience in the context of sparse coding in the mammal visual system [Olshausen and Field (1996)], sparsity has become a key concept of fundamental importance for many fields including machine learning, statistics and signal processing [Tibshirani (1996); Chen et al. (2001); Candès and Wakin (2008)]. Based on the applications in different research fields, there exist several commonly-used sparsity measures [Hurley and Rickard (2009)]. Here, we present the sparseness measures used for the analysis purposes in the following chapters.

The $\ell_0$-norm measure is the traditional sparseness measure in many mathematical settings. It simply calculates the number of non-zero entries in the representation vector $v$. However, the presence of noise makes the $\ell_0$-norm measure completely inappropriate (since it relies on hard zeros). Consequently, in noisy settings, the $\ell_0$-norm measure is sometimes altered to $\ell_0^\epsilon$, where we are interested in the number of coefficients, $v_i$ that are greater than the threshold $\epsilon$ [Rath et al. (2008)]. Evidently, the value of $\epsilon$ is crucial for $\ell_0^\epsilon$ to be effective.

Hoyer measure [Hoyer (2004)] is the normalized version of the $\ell_2/\ell_1$ measure. It generates a score in $[0-1]$, where 0 denotes the least sparse case (i.e., uniform distribution of energy

among the coefficients) and 1 denotes the most sparse case (i.e., spiky distribution, almost all energy is carried by only one coefficient). It is formulated in (3.10), with $v$ denoting the representation vector with dimension $n$.

$$\text{sparseness}(v) = \frac{\sqrt{n} - (\sum |v_i|) / \sqrt{\sum v_i^2}}{\sqrt{n} - 1} \tag{3.10}$$

It is important to note that the increase in the sparsity based on Hoyer measure does not necessarily mean that the coefficients $v_i$ (i.e., entries in the representation vector $v$) are hard-zeros. In other words, sparse activity based on Hoyer measure does not imply sparse coding.

## 3.3 Conclusion

In this chapter, we provided the background information about autoencoders, with special emphasis on sparse autoencoders. There are different model configurations under the name of sparse autoencoder due to the lack of consensus in literature regarding its definition. To simplify, we categorized the sparse autoencoders whether they are explicitly or implicitly constrained. The penalty term is added to the loss function for the case of explicitly constrained sparse autoencoders. Whereas, the sparsity on the model is maintained by internal sparsity mechanisms for the case of implicitly constrained sparse autoencoders.

Shallow overcomplete sparse autoencoder with $\ell_1$ norm regularization on the hidden unit activations (3.6) constitutes the most intuitive example for explicitly constrained sparse autoencoders, due to its often over-looked connection with sparse coding and dictionary learning concepts. In Chapter 4 and Chapter 5, we use this sparse autoencoder configuration for sparse modeling of speech data.

We observe the limitations of shallow overcomplete sparse autoencoder for speech modeling; hence, we use implicitly constrained sparse autoencoders in Chapter 6. We use k-Sparse autoencoder [Makhzani and Frey (2013)] which is the most well-known example for implicitly constrained sparse autoencoders. However, k-Sparse autoencoder may suffer from dead atom problems (Section 3.2.3).

As we seek biologically motivated, simple yet elegant way of learning meaningful representations in an unsupervised and cost effective manner, we studied the sparsity and different viewpoints to sparsity (i.e., population sparseness, life-time sparseness) in Section 3.2.4.

We subsequently study Winner-Take-All (WTA) autoencoder [Makhzani and Frey (2015)] (in Section 6.3), which adopts life-time sparseness (Figure 3.5) to avoid dead atom problem. Thus, this model can more easily learn Sparse Distributed Representations (SDRs), that are meaningful and discriminative by nature. This behavior is expected to be useful for encoding

unseen data points reasonably (i.e., out of distribution generalization). Furthermore, as shown in Chapter 6, it is convenient for classifiers to reach higher accuracy while exploiting the correlations between the representations in the supervised learning case.

Hence, in Chapter 7, we examine the generalization power of the best performing implicitly constrained sparse autoencoder (trained on healthy speakers) from Chapter 6 in transfer learning framework for pathological speech recognition.

# 4 Low-Rank and Sparse Modeling of LF-MMI Log-Likelihoods

In this chapter [1], we introduce our proposed approach which makes use of undercomplete autoencoders for low-rank modeling and sparse overcomplete autoencoders for sparse modeling of log-likelihoods from Lattice-Free Maximum Mutual Information (LF-MMI) based acoustic model.

In summary, we insert an additional autoencoder component between the acoustic model and decoder components in the ASR pipeline (Figure 2.1). Upon autoencoder training, the reconstructed LF-MMI log-likelihoods are sent to the decoder for recognition. Our proposed sparse modeling approach (with sparse overcomplete autoencoder) is observed to improve the recognition performance for far-field speech.

Similarly, the sparse modeling approach yields better at representing speech data as a union of low-dimensional subspaces, especially for far-field speech. The encodings learned by the sparse autoencoders are used to train phone classifiers for frame-level phone classification. As a result, we observe superior performance of encodings from sparse overcomplete autoencoder for far-field data.

The analysis on high-dimensional sparse features (i.e., encodings from sparse overcomplete autoencoder) verifies their sparseness based on the measures in Section 3.2.5, and demonstrates the capability of robust modeling of senones and even phone subspaces, especially on far-field speech with distorted nature.

## 4.1 Introduction

Given the underlying speech production mechanism, one can assume that speech can be described as the union of low-dimensional subspaces. To model such phenomena, [Bengio (2009)] outlines two forms of modeling, (1) as compressed low-rank representations for each

[1] This chapter is partially based on the following publications:
Kabil, S.H., and Bourlard, H. (2022). Speech modeling using sparse autoencoders.
Kabil, S.H., and Bourlard, H. (2022). Sparse autoencoders to enhance speech recognition.

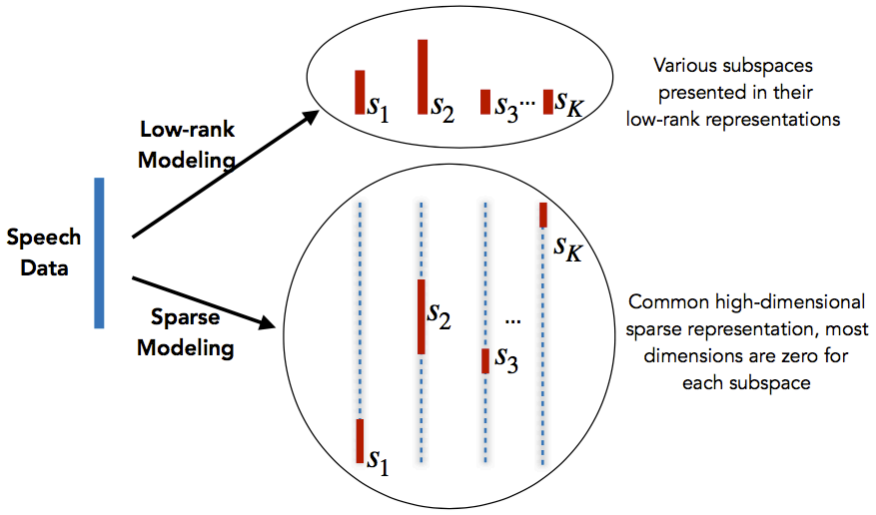



***Figure 4.1:*** *Modeling class-specific low-dimensional subspaces in speech data using low-rank vs sparse modeling approach in [Dighe (2019)]. Note that the low-rank modeling corresponds to class-wise low-rank models, not a common low-rank model for the whole (multi-class) speech data together. $s_k$ denotes $k^{th}$ subspace in the data. Figure reproduced from [Dighe (2019)] with permission.*

factor's subspace individually, and (2) as a high-dimensional sparse representation where all the factors reside together, but in different subspaces. Low-rank representations model the subspaces in a compressed manner by eliminating irrelevant dimensions of the data, whereas sparse representations achieve the same objective by projecting data in a high-dimensional sparse space where the underlying structures are disentangled, and only the relevant dimensions are non-zero (i.e., active).

To this end, [Dighe (2019)], which serves as the backbone of the research in this thesis, investigates the impact of low-rank and sparse modeling of DNN acoustic model outputs (i.e., posterior features) for speech recognition. Typically, DNN posteriors are high-dimensional with only a few non-zero dimensions which correspond to different low-dimensional subspaces covering underlying factors like acoustic events (i.e., realization of particular phonetic sounds or pronunciation variations due to speaker characteristics). In [Dighe (2019)], it is hypothesized that appropriate modeling of low-dimensional subspaces in DNN posteriors can result in improvements in recognizing words, phonemes and sub-phonetic components.

For low-rank modeling, ground-truth based forced senone alignments are required. Posterior vectors belonging to a particular senone are stacked together to form a senone-specific posterior matrix. The principal components learned using Principal Component Analysis (PCA) on these posterior matrices act as senone-specific (i.e., class-specific) undercomplete dictionaries. As shown in Figure 4.1, when a posterior vector is projected on its corresponding undercomplete dictionary, densely packed, class-wise low-dimensional representations are

extracted.

For sparse modeling, the undercomplete senone-specific dictionaries from low-rank modeling approach are concatenated to form an initialization for the overcomplete dictionary, which is later trained via online dictionary learning algorithm [Mairal et al. (2009)]. The overcomplete dictionary is capable of modeling non-linear speech manifolds as a union of low-dimensional spaces. Hence, when posterior features are projected over the overcomplete dictionary, senone-specific sparse representations manifest themselves on different dimensions in the common high dimensional space, as shown in Figure 4.1.

Then, these intermediate representations (shown in red in Figure 4.1) extracted either by low rank (PCA based) or sparse (dictionary learning based) modeling are projected back onto the original dimensions of the DNN posterior space. The reconstructed enhanced DNN posteriors are shown to be suitable targets for training better acoustic models, leading to improvements in recognition performance.

As both above-mentioned approaches require senone alignments, for unseen test data (without transcriptions and senone alignments), a completely new DNN acoustic model is trained using acoustic features as input and enhanced DNN posteriors as soft targets. After training, test data is fed to the new acoustic model to get posteriors for ASR decoding.

The experiments on the AMI database show that dictionary learning based soft targets are observed to be better than their PCA counterparts for SDM (far-field) data, which is contrary to their performance on IHM (close-field) data. The success of sparse soft targets for SDM indicates that the nonlinear low-dimensional modeling of senone subspaces while suppressing distortions is better handled by sparse modeling.

Accordingly, our primary focus in this thesis is sparse modeling (of speech data) in a generic and unsupervised manner, by means of sparse autoencoders. Low rank modeling experiments with undercomplete autoencoders are presented for the sake of consistency with previous studies.

## 4.2 Our Approach

As illustrated in Figure 2.1, we aim here at inserting a separately trained autoencoder (AE) between the acoustic model and the decoder components in the ASR pipeline. The low rank and sparse modeling of (senone) log-likelihoods from the previously trained LF-MMI baseline acoustic model (Section 2.5) are extracted by undercomplete (Figure 4.2) and sparse overcomplete (Figure 4.4) autoencoders, respectively.

After the usual (unsupervised) AE training, the reconstructed (lower-rank) log-likelihood vectors are fed to the decoder for recognition (illustrated with blue arrow in Figures 4.2 and 4.4). The resulting word recognition performance in Word Error Rate (WER) are presented in Tables 4.1 and 4.2.

In addition, (again, after AE training), simple DNN phone classifier is trained on the hidden unit activations (i.e., AE encodings, embeddings) to assess the representational capacity of these intermediate features (illustrated with red arrow in Figures 4.2 and 4.4). The frame-level phone accuracies are reported in Table 4.3.

### 4.2.1 Low-Rank Modeling of LF-MMI Log-Likelihoods

In Figure 4.2, $\mathbf{x}$ represents the (senone) log-likelihood vector with dimension 176 and $\tilde{\mathbf{x}}$ represents its reconstructed version. As we use undercomplete AE, the encoding $\mathbf{z}$ is lower in dimension than $\mathbf{x}$.

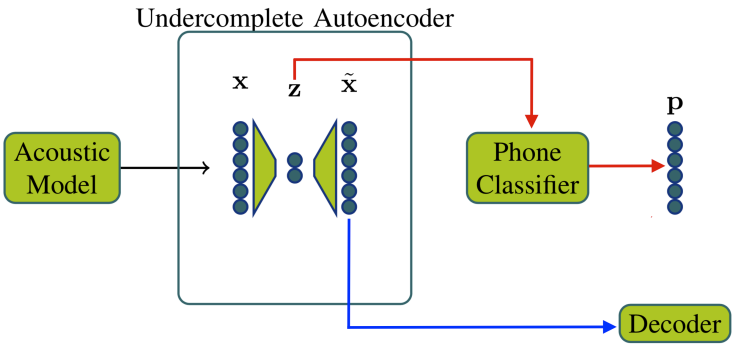


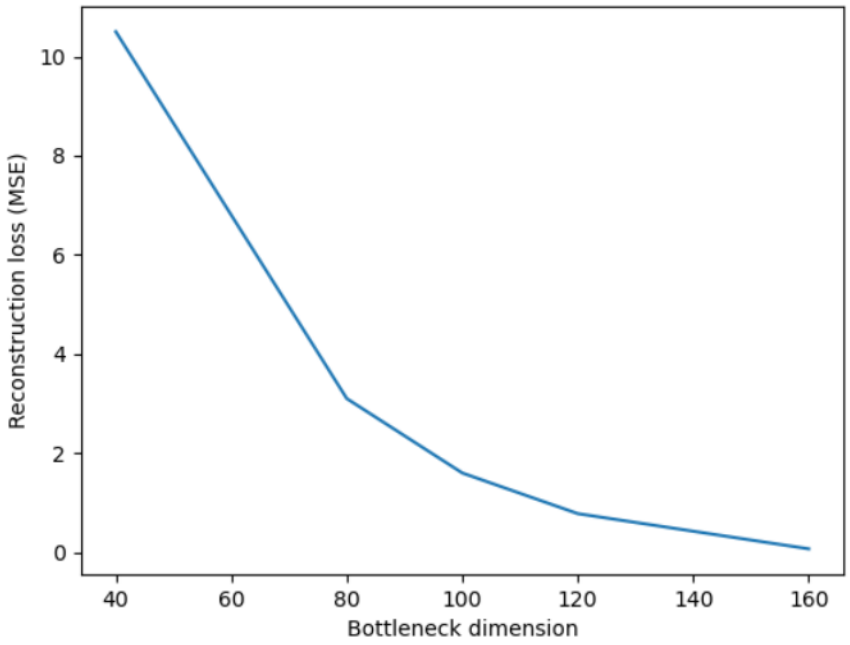

***Figure 4.2:*** *Low-rank modeling of LF-MMI log-likelihoods by means of undercomplete AE. The reconstructed log-likelihood vectors are sent to the decoder for word recognition. The performance (in WER %) is presented in Table 4.1. In addition, simple DNN phone classifiers are trained to assess the representation power of the AE encodings. The frame-level phone accuracies are reported in Table 4.3.*

To determine the optimal dimension of $\mathbf{z}$ in autoencoders (trained) on close-field IHM and far-field SDM speech, we examine the elbow region on development sets (Figure 4.3). We set the bottleneck dimension for $\mathbf{z}$ as 100 and 120 for undercomplete AE trained on IHM and SDM, respectively.

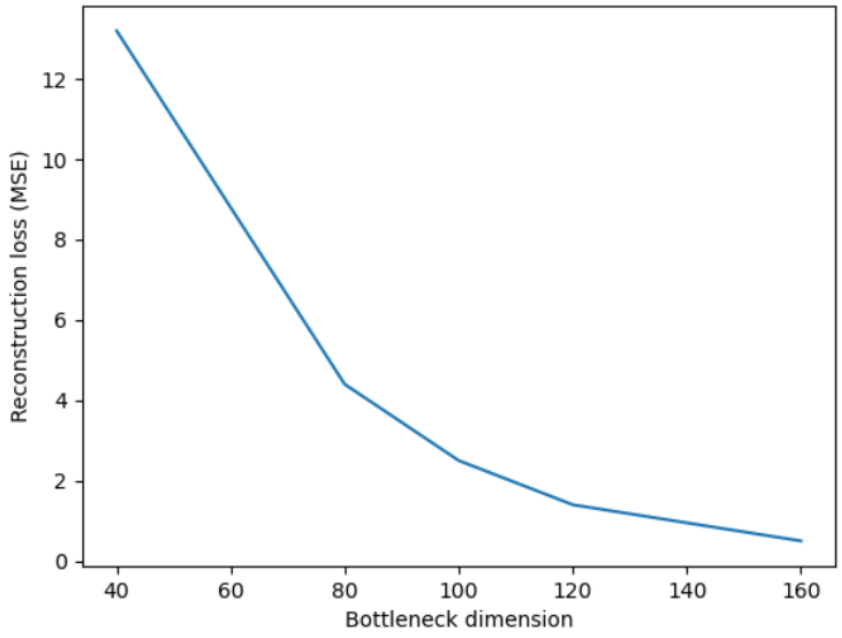
The autoencoders are implemented in Pytorch [Paszke et al. (2017)]. Note that all other components are implemented in Kaldi [Povey et al. (2011)] speech processing toolkit. For reading and writing Kaldi data format (e.g., ark files), we use a pure Python module called *kaldiio* [2].

The log-likelihoods are preprocessed by subtracting each frame with its largest value (i.e., mapping the log-likelihood frames to $(-\inf, 0]$ region). We observe that this normalization scheme helps with machine accuracy and autoencoder training. All autoencoders are trained for (maximum of) 100 epochs with Adam optimizer and scheduler for early stopping with patience of 5 epochs.

[2] https://github.com/nttcslab-sp/kaldiio




**(a)** IHM




**(b)** SDM

***Figure 4.3:*** *Optimal bottleneck dimension for undercomplete AEs are determined by observing the elbow region (L-curve) on the development set. The bottleneck dimension is set 100 and 120 for undercomplete AE trained on IHM and SDM, respectively.*

After AE training, the reconstructed log-likelihoods are sent to the Kaldi decoder for ASR decoding. As we put our autoencoder component between the LF-MMI acoustic model and the decoder, we can no longer use *nnet3-latgen-faster* command in Kaldi for decoding graph generation. Instead, we use *nnet3-compute | latgen-faster-mapped* command sequence. In any way, Kaldi decoder expects log-likelihood features as input. During experiments, we notice that training the autoencoders with posterior features (i.e., applying exp() on log-likelihoods from the acoustic model) and then mapping them back to the log domain before feeding the Kaldi decoder, results in degradation on WER. Thus, we stick to the log-likelihoods as input features to train the AEs in this chapter.

***Table 4.1:*** *The recognition performance (in WER%) for baseline LF-MMI system and the proposed approach on IHM and SDM evaluation sets. Low-rank modeling setup with undercomplete AE works better for IHM than SDM. This trend is consistent with the findings presented in [Dighe et al. (2019)].*

| Architecture | $IHM_{WER}$ | $SDM_{WER}$ |
|---|---|---|
| Baseline | 19.2 | 41.1 |
| Undercomplete AE | 19.1 | 41.4 |

In other words, instead of passing the log-likelihoods from the LF-MMI acoustic model directly to the decoder (indicated as baseline in Tables 4.1 and 4.2), we send their reconstruction from the undercomplete AE to the decoder (indicated as "Undercomplete AE" in Table 4.1). We observe that low-rank modeling setup works better for IHM than SDM. This trend is consistent with the findings from [Dighe et al. (2019)].

### 4.2.2 Sparse Modeling of LF-MMI Log-Likelihoods

As previously explained, our goal is to achieve sparse modeling of speech data in a generic and unsupervised manner by means of sparse autoencoders. In [Dighe (2019)], sparse modeling approach is observed to be better at modeling speech data as union of low-dimensional subspaces, especially for SDM in far-field condition.

Revisiting the duality between dictionary learning [Mairal et al. (2009)] and sparse autoencoders, we implement shallow overcomplete sparse autoencoder with $\ell_1$ norm constraint on hidden activations with loss function formulated as (3.6). In addition, to ensure full compliance with dictionary learning (3.7), the encoder and decoder weights are tied (i.e., transpose of each other), the encoder weights are normalized, and no bias parameter is used.

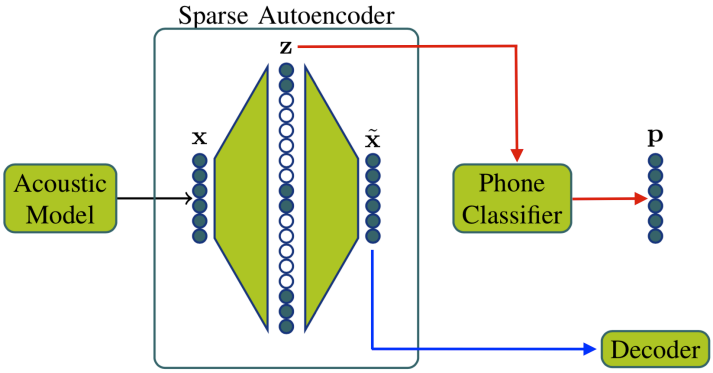


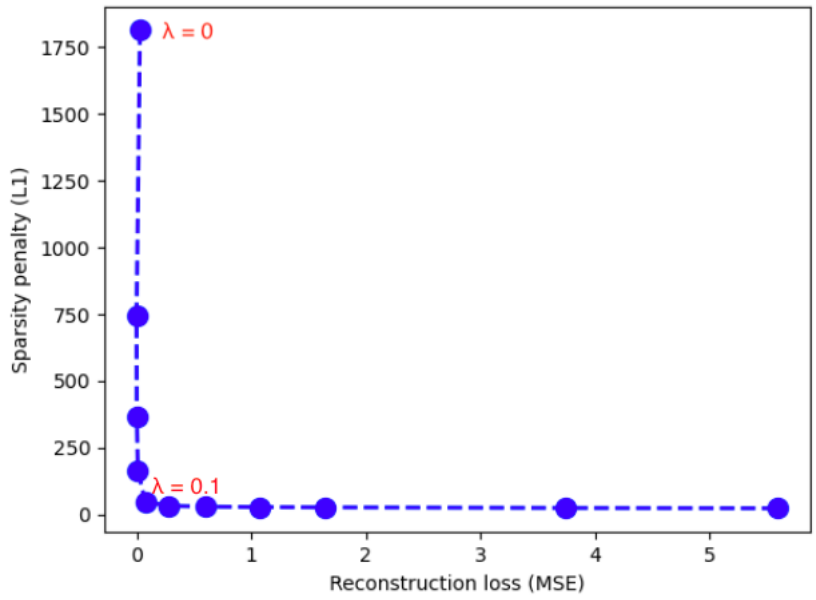

***Figure 4.4:*** *Sparse modeling of LF-MMI log-likelihoods by means of sparse overcomplete AEs. The reconstructed log-likelihoods are sent to the decoder for recognition. The performances (in WER%) are presented in Table 4.2. Furthermore, simple DNN phone classifiers are trained on AE encodings to assess their representation power. The frame-level phone accuracies are reported in Table 4.3.*

Hence, we effectively convert the dictionary learning problem to a representation learning problem. When the sparse AE is trained to solve (3.6), forward pass can be viewed as the sparse coding step in dictionary learning, since we obtain high-dimensional sparse representation (i.e., encoding) $\mathbf{z}$ on the hidden layer. Similarly, the backward pass is analogous to the dictionary update step in dictionary learning, as decoder weights (akin to the overcomplete dictionary in dictionary learning setup) are updated based on the distance between the original input $\mathbf{x}$ and reconstructed input $\tilde{\mathbf{x}}$. In Figure 4.4, $\mathbf{x}$ represents the LF-MMI (senone) log-likelihood vector with the dimension of 176, and $\mathbf{z}$ denotes the overcomplete encoding vector with dimensionality of 1760 (i.e., 176 x 10, where 176 is the number of senones in Kaldi AMI setup and the scalar 10 is empirically determined so that log-likelihoods are given chance to scatter).

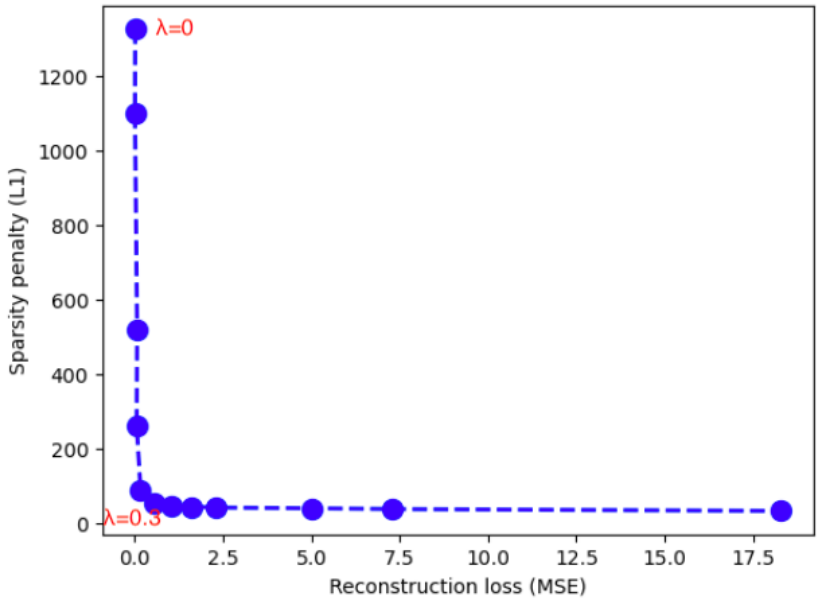
To determine the optimal $\lambda$ (in 3.6) for weighting the sparsity penalty term, we follow the L-curve on development sets, formed by sparsity penalty (i.e., unweighted $\ell_1$ sum) versus reconstruction loss (i.e., MSE), as illustrated in Figure 4.5. The optimal $\lambda$ coefficients for IHM and SDM are 0.1 and 0.3, respectively.

In full accordance with the implementation details in Section 4.2.1, the autoencoders are implemented in Pytorch, following the same preprocessing and training procedure. After sparse AE training, reconstructed log-likelihoods are sent to the Kaldi decoder and *nnet3-compute | latgen-faster-mapped* command series is used for generating lattices (and then for decoding graph in Kaldi).

In other words, instead of passing the LF-MMI log-likelihoods from the LF-MMI acoustic model directly to the decoder (indicated as baseline in Table 4.1 and 4.2), we pass the reconstructed log-likelihoods from the shallow sparse overcomplete autoencoder to the decoder (indicated as Sparse AE in Tables 4.2).




**(a)** IHM




**(b)** SDM

***Figure 4.5:*** *Optimal sparsity penalty coefficients for sparse overcomplete AEs are determined by observing the elbow region (L-curve) on development set, formed by sparsity penalty (i.e., unweighted $\ell_1$ sum) versus reconstruction loss (i.e., MSE). The optimal $\lambda$ coefficients for IHM(a) and SDM(b) are set 0.1 and 0.3, respectively.*

***Table 4.2:*** *The recognition performance (in WER%) for the baseline LF-MMI system and the proposed approach on IHM and SDM evaluation sets. For the sake of completeness, the results for Vanilla (λ=0) overcomplete AE are also reported. In line with the findings in [Dighe (2019)], our proposed sparse modeling approach with sparse overcomplete autoencoder (denoted as Sparse AE) works better for far-field SDM data than close field IHM.*

| Architecture | $IHM_{WER}$ | $SDM_{WER}$ |
|---|---|---|
| Baseline | 19.2 | 41.1 |
| Overcomplete AE ($\lambda = 0$) | 19.2 | 41.0 |
| Sparse AE | 19.2 | 40.9 |

For the sake of completeness, we also report the results for vanilla (i.e., $\lambda = 0$) overcomplete AE. In line with findings in [Dighe (2019)], sparse modeling approach works better for far-field SDM data (i.e., improvement on WER with overcompleteness and sparsity) than close-field IHM, as shown in Table 4.2. However, it is worth to mention that our state-of-the-art baseline system is more complex (in terms of number and composition of layers, MMI criterion etc.) and challenging than the baseline system in [Dighe (2019)] with 3-5 fully-connected layers trained with CE-criterion. Hence, it is very likely that our LF-MMI acoustic model produces cleaner, more discriminative log-likelihoods on which we still try to further improve.

## 4.3 Frame-level Phone Accuracy

Apart from exploring the use of reconstructed log-likelihoods for speech recognition (Section 4.2), we also examine the practical application of AE encodings (previously denoted as $z$) in the same context. Nonetheless, aforementioned requirements imposed by the Kaldi decoder restrict the use of these encodings for word recognition. Therefore, we train simple DNN classifiers on these features for frame-level phone classification tasks.

To evaluate the efficiency of the features, but not the classifier, we keep the classifier configuration basic with only fully connected layers. The classifiers are implemented using Pytorch, and trained with cross-entropy (CE) criterion. We use *ali-to-phones* command with *–per-frame=true* flag in Kaldi to extract the forced (frame-level) phone alignments. These alignments are then used as labels for the classifier.

***Table 4.3:*** *The frame-level phone classification accuracies (in %) on IHM and SDM evaluation sets. The encodings from sparse overcomplete AE (denoted as Sparse AE) obtain superior performance for far-field SDM data, which is known to contain more distortion than close-field IHM data.*

| Architecture | $IHM_{accuracy}$ | $SDM_{accuracy}$ |
|---|---|---|
| Baseline | 82.7 | 61.0 |
| Undercomplete AE | 81.5 | 58.6 |
| Overcomplete AE ($\lambda = 0$) | 82.3 | 60.8 |
| Sparse AE | 82.5 | 63.7 |

We observe that encodings from sparse overcomplete AE provide superior performance for far-field SDM data in frame-level phone classification task. This might indicate the representational power of these high-dimensional sparse features in the presence of distortion. Therefore, in the next section, we analyze whether they are indeed sparse and meaningful, as expected.

## 4.4 Analysis on High-dimensional Sparse Features

As shown in Table 4.3, we observe that high-dimensional encodings from sparse AE yield superior performance for frame-level phone classification, especially for far-field SDM data which is known to have more distortions than close-field IHM.

In this section, we first examine if the $\ell_1$ norm sparsity constraint (imposed on sparse AE) behaves as expected and guides the AE to produce high-dimensional sparse encodings. Along this line, to evaluate the sparsity of hidden unit activations (i.e., encodings), we utilize $\ell_0$, $\ell_0^\epsilon$ and Hoyer measure (introduced in Section 3.2.5).

Then, in Section 4.4.2, we investigate if the encoding space (i.e., representation space) exhibits orthogonalization or smooth latent space properties (Section 3.2.2) when cosine similarity is used as similarity measure.

### 4.4.1 Sparsity of Activations

As previously explained in Section 3.2.1, the definition for sparseness may differ. Here, we examine if the hidden unit activations (i.e., AE encodings) are indeed sparse, in the sense that **(1)** they contain hard-zeros, or **(2)** only a few dimensions of the activation vector (i.e., encoding vector) carry most of the information content (while none of the dimensions are not necessarily hard-zero).

In shallow sparse AE (3.6), we apply $\ell_1$ norm penalty on hidden unit activations, which pulls all the dimensions (i.e., entries, coefficients) of the activation vector towards zero. Hence, for **(1)**, we utilize $\ell_0$ and $\ell_0^\epsilon$ measures (Section 3.2.5). Based on $\ell_0$ measure, we observe that we do not obtain hard-zero activations (shown in Tables 4.4 and 4.5). On the other hand, $\ell_0^\epsilon$ with $\epsilon$ set according to the value range on development sets, we see that sparsity constraint helps to pull the activations towards zero, although fails to produce hard-zero activations as expected (illustrated in Tables 4.4 and 4.5, second column).

Whereas, for **(2)**, we utilize Hoyer measure [Hoyer (2004)], where in $v$ with dimension $n$ in ( 3.10) stand for the hidden unit activation vector $z$ in Figure 4.4, with dimension 1760, respectively. It is important to note that sparseness based on Hoyer measure does not necessarily mean that the entries (i.e., dimensions, coefficients) of the activation vector (i.e., encoding vector, representation vector) are hard-zero. Hoyer measure generates a score in $[0,1]$ where 0 denotes the least sparse case (i.e., uniform distribution of information among entries of

***Table 4.4:*** *The sparsity of the hidden unit activations based on the complements of $\ell_0$ and $\ell_0^\epsilon$ measures for a chosen subset of utterances from IHM development set. For simplicity, instead of a total number of zero entries, we provide their percentage over all entries in the activation vectors for (randomly chosen) subset of utterances. We observe that the sparsity constraint ($\lambda > 0$, optimally being 0.1 for IHM) pulls hidden unit activations towards zero, although fails to generate hard-zero activations.*

| Architecture | $(\ell_0)^c$ (in %) | $(\ell_0^\epsilon)^c$ (in %) |
|---|---|---|
| IHM ($\lambda = 0$) | 0 | 10 |
| IHM ($\lambda = 0.1$) | 0 | 20 |

***Table 4.5:*** *The sparsity of the hidden unit activations based on the complements of $\ell_0$ and $\ell_0^\epsilon$ measures on (a randomly chosen) subset of utterances from SDM development set. For simplicity, instead of a total number of zero entries, we provide their percentage over all entries in the activation vectors for (randomly chosen) subset of utterances. We observe that the sparsity constraint ($\lambda > 0$, optimally being 0.3 for SDM) pulls hidden unit activations towards zero, although fails to generate hard-zero activations.*

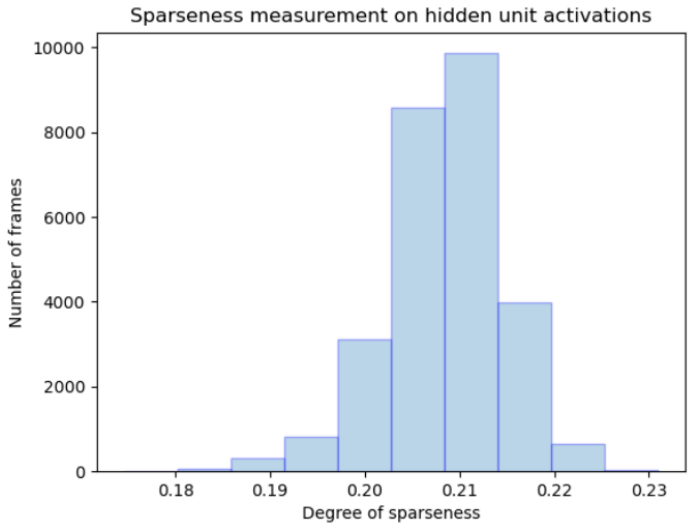
| Architecture | $(\ell_0)^c$ (in %) | $(\ell_0^\epsilon)^c$ (in %) |
|---|---|---|
| SDM ($\lambda = 0$) | 0 | 13 |
| SDM ($\lambda = 0.3$) | 0 | 21 |

the hidden unit activation vector) and 1 denotes the most sparse case (i.e., spiky distribution, almost all information is carried by only one coefficient).

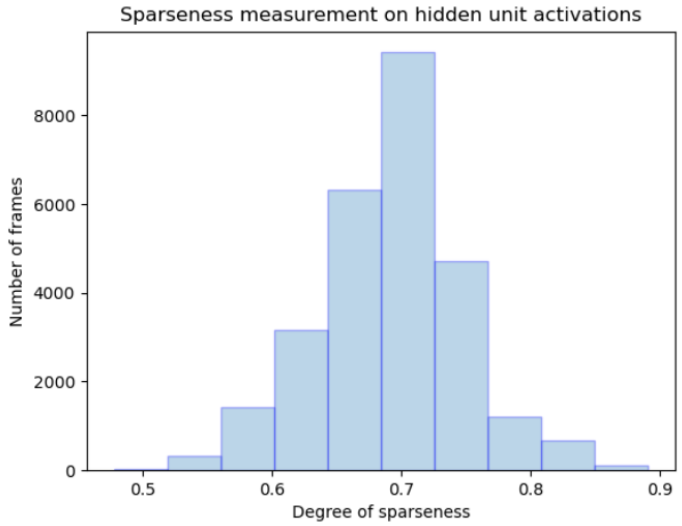
Focusing on the *same* randomly chosen subset of utterance from development sets, we calculate the sparseness of the hidden unit activations, with Hoyer measure. Figures 4.6 and 4.7 for sparseness measurement of the hidden unit activation vectors for (a) Vanilla ($\lambda$=0) overcomplete and (b) Sparse overcomplete AE on IHM and SDM, respectively. *x* axis shows the Hoyer score, *y* stands for the number of activation vectors. It is important to note that each (input) log-likelihood vector (i.e., frame) has a corresponding activation vector revealed on the AE's encoding layer. Hence, histograms clearly depict the alteration in the characteristics of the activations (i.e., encodings) in the presence of sparsity.

Table 4.4, Figure 4.6 (for IHM) and Table 4.5, Figure 4.7 (for SDM) are computed on the *same* subset of utterances from development sets. Therefore, they present a complementary view for the analysis of the sparsity of activations. In conclusion, we observe that $\ell_1$ norm sparsity penalty (on hidden units) guides the autoencoder to force its hidden units to produce close-to-zero activations. In addition, while the majority of the hidden units produce close-to-zero activations, a small number of hidden units produce the majority of the activation (i.e., information, energy), forming the rare spikes among many close-to-zero entries in the activation vector.

In the rest of the analysis, we examine if these spiky hidden unit activations actually represent meaningful patterns, modeling senone (or any other speech unit's) subspaces.

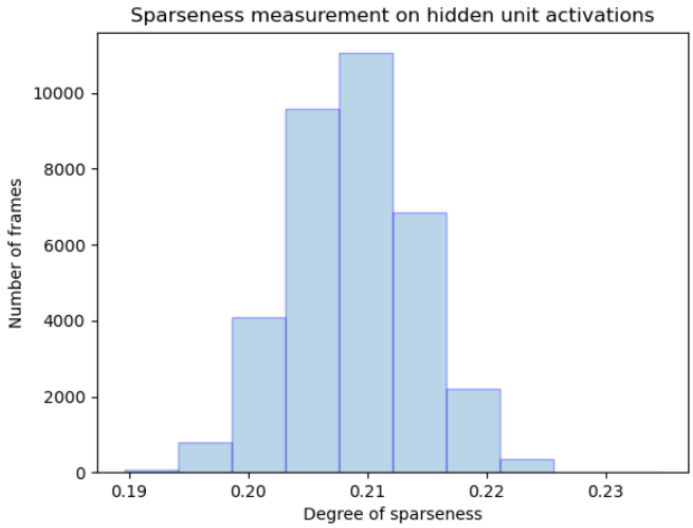



**(a)** IHM $\lambda$=0

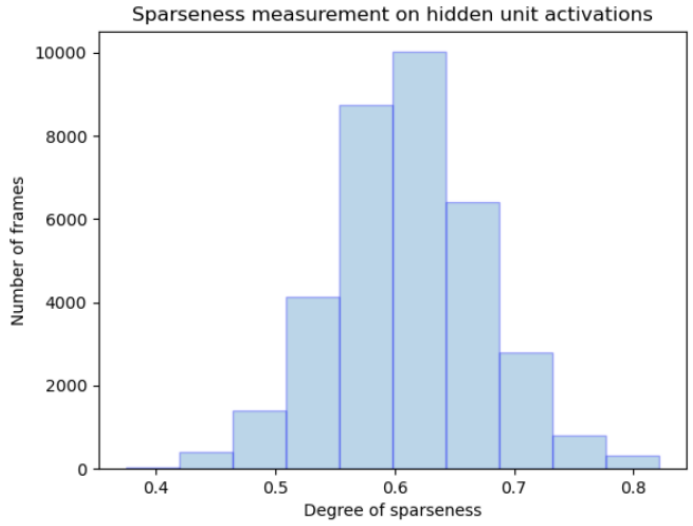



**(b)** IHM $\lambda$=0.1

***Figure 4.6:*** *Sparseness measurement on the hidden unit activations for randomly chosen subset of utterances from IHM development set. Enforcing sparsity ($\lambda > 0$) is observed to change the characteristics of the activations. For Vanilla ($\lambda$=0) overcomplete AE (a), majority of the activation vectors have 0.2; whereas, for sparse overcomplete AE (b), majority of the activation vectors have 0.7 as Hoyer sparseness score. On the scale of* $[0,1]$*, 0 denotes the least sparse, 1 denotes the most sparse case.*




**(a)** SDM $\lambda$=0




**(b)** SDM $\lambda$=0.3

***Figure 4.7:*** *Sparseness measurement on the hidden unit activations for randomly chosen subset of utterances from SDM development set. Enforcing sparsity ($\lambda > 0$) is observed to change the characteristics of the activations. For Vanilla ($\lambda$=0) overcomplete AE (a), majority of the activation vectors have 0.21; whereas, for sparse overcomplete AE (b), majority of the activation vectors have 0.63 as Hoyer sparseness score. On the scale of* $[0,1]$*, 0 denotes the least sparse, 1 denotes the most sparse case.*

### 4.4.2 Subspace Analysis

With our proposed approach for sparse modeling, we aim for individual subspaces to lie in the common high dimensional representation space in a scattered manner (as shown in Figure 4.1). Our hypothesis is that this behavior can induce the separability of senones (and even phones).

Based on the findings in Section 4.4.1, we anticipate the hidden unit activation vectors for a particular senone to have a distinctive pattern and to share a few common features with other senones that have similar phonetic and articulatory characteristics. Note that senones are formed by clustering the phones based on the decision tree questions regarding contextual and phonetic properties. That is why, later in our analysis here, we take the phones as the disentangled form of senones.

Hence, focusing on the *same* subset of utterances in Section 4.4.1 (with 419 utterances and 112386 frames), we average all activation vectors belonging to a particular senone id (i.e., class) and set the resulting vector as *fingerprint* for that particular senone. Inspired from sparse distributed memory [Kanerva (1988)], we assume that fingerprint vectors lie in a vector space (i.e., representation space) with cosine similarity (5.1) as similarity metric. Shortly, we hypothesize that we observe smooth latent space properties (Section 3.2.2) to emerge on the vector space where the fingerprint vectors (for all seen/observed senones in the subset) reside.

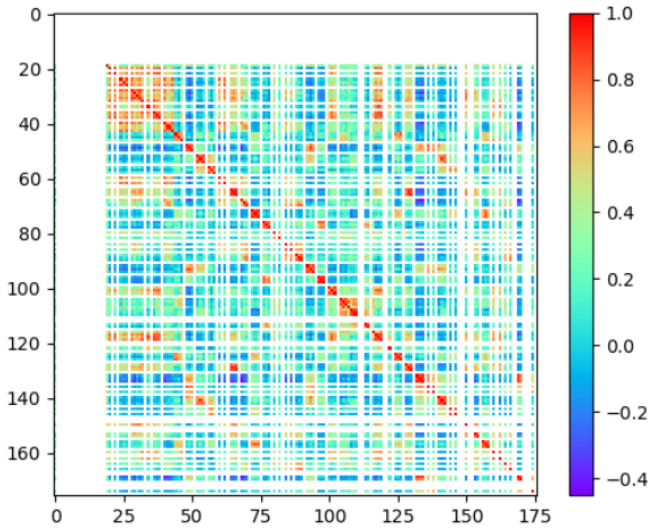
$$\text{similarity } = \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}, \tag{4.1}$$

Figures 4.8 and 4.9 illustrate the similarities between the observed senone classes in the subset among overall 176 senones in the setup. Similarity scores are computed via cosine similarity metric (4.1). Cosine similarity produces scores $[-1, 1]$, 1 denotes the highest similarity (shown in red in figures).

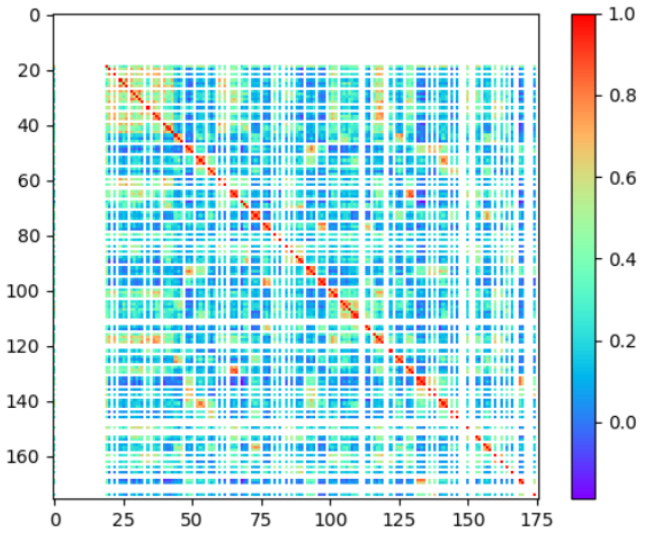
Thanks to these colormaps, we make two important observations which motivate the further analysis:

1. We observe that even Vanilla AEs (with $\lambda = 0$) are able to capture similarity patterns among senones. In addition, the legends (of the colormaps) indicate the impact of sparsity constraint. As shown in Tables 4.4 and 4.5, activation values are approaching to zero (i.e., being small $\epsilon$ value); hence, their product during cosine similarity score computation also converges to zero, shrinking the range of the colormap region.

2. As a result, we observe that senones with strongest similarity scores (shown in red) become prominent for both IHM and SDM data (Figures 4.8(b), 4.9(b)).

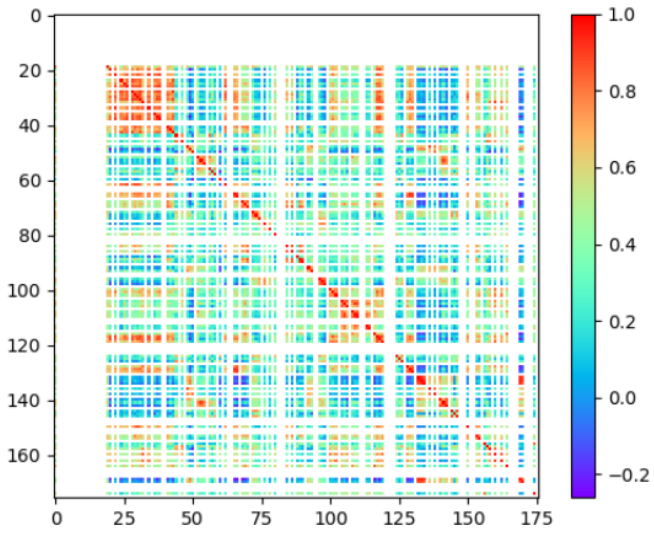



**(a)** IHM $\lambda$=0

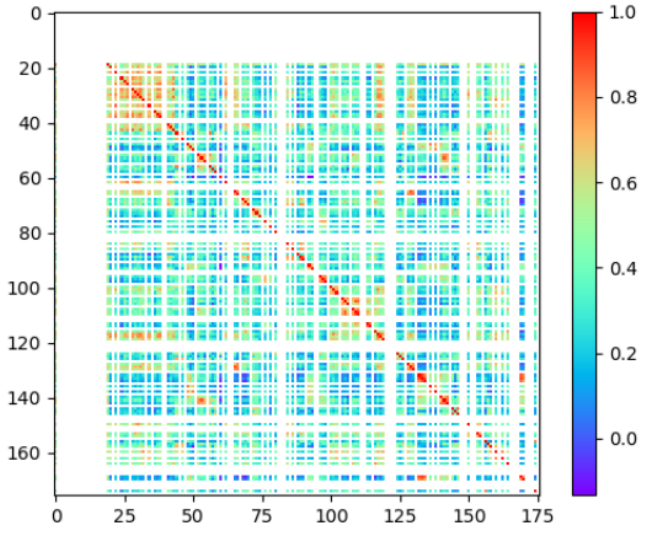



**(b)** IHM $\lambda$=0.1

***Figure 4.8:*** *Colormaps for visualizing the similarity analysis between the senone-specific fingerprint vectors. Note that not all 176 senones (in the setup) occur in the subset of utterances, which is reflected as white regions in the colormaps above. The similarities between the (fingerprints of) observed senones are computed with cosine similarity. We observe that in the presence of sparsity ($\lambda > 0$), the similarity between senones with high cosine similarity (close to 1, shown in red) becomes more prominent. This can also be traced in the shrinkage on the value range of the colormap legends (i.e.,* $[-0.4, 1]$ *for* $\lambda = 0$*, whereas* $[0, 1]$ *for* $\lambda = 0.1$*).*




**(a)** SDM $\lambda$=0




**(b)** SDM $\lambda$=0.3

***Figure 4.9:*** *Colormaps for visualizing the similarity analysis between the senone-specific fingerprint vectors. Note that not all 176 senones (in the setup) occur in the subset of utterances, which is reflected as white regions in the colormaps above. The similarities between the (fingerprints of) observed senones are computed with cosine similarity. We observe that in the presence of sparsity ($\lambda > 0$), the similarity between senones with high cosine similarity (close to 1, shown in red) becomes more prominent. This can also be traced in the shrinkage on the value range of the colormap legends (i.e., $[-0.2, 1]$ for $\lambda = 0$, whereas $[0, 1]$ for $\lambda = 0.3$).*

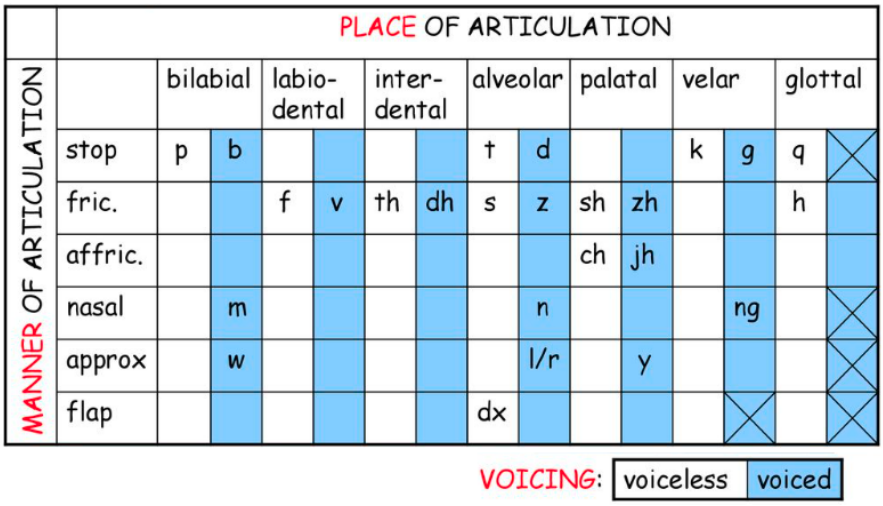



***Figure 4.10:*** *Articulatory parameters for English consonants in ARPAbet [Vendetti (2002a)]. The matches between fingerprint vectors mapping to similar phone labels in terms of manner and/or place of articulation are taken as robust match. The details of the analysis whose goal is to investigate whether the representation space (where fingerprint vectors reside) exhibits orthogonalization and/or smooth latent space properties (Section 3.2.2) can be found in Tables 4.6 and 4.7. The manifestation of these properties is important for understanding the representation capacity of the AE encodings.*

Based on our findings with colormaps, we extend our subspace analysis with fingerprint vectors. Thanks to Kaldi's *ali-to-pdf*, *ali-to-phones*, and *show-transitions* commands, we manage to generate mappings between the senones (i.e., pdfs in Kaldi terminology) and phones with positional encodings. We hypothesize that fingerprint vectors can also shed light on the phone-level patterns which are in more human interpretable form.

Using Figures 4.10 and 4.11, we take the matches between fingerprint vectors, mapping to similar phone labels in terms of manner and/or place of articulation, as *robust* match. For instance, given cosine similarity as metric and a matching threshold, the match between the fingerprints for /s/(fricative, alveolar) and /z/(fricative, alveolar) is robust. On the other hand, the match between the fingerprints for /s/ and /m/(nasal, bilabial) is a false match (i.e., not robust match); hence, not desired, especially when the matching threshold is decreased.

We examine the ratio of number of *robust* matches over total matches with respect to the changes in the matching threshold, $\lambda$ and data (i.e., on which fingerprint vectors are computed). As shown in Tables 4.6 and 4.7, with the decrease in threshold, the number of total matches increases faster than the number of robust matches (i.e., the ratio decreases). The increase in $\lambda$ amps up the ratio of robust matches over all matches.

***Table 4.6:*** *The ratio of number of robust matches over the total number of matches (in [0-1] scale) with respect to the changes in the matching threshold, $\lambda$, and data. With decrease in matching threshold, we expect this ratio to get closer to 0 (as the total number of matches is expected to increase swiftly), unless the model has robust modeling power to stabilize the ratio of robust matches over all matches. The changes on $\lambda$ do not create any significant changes for the number of robust matches on IHM data, compared to SDM data. Note that the robust matches are determined based on the tables presented in Figures 4.10 and 4.11.*

| Threshold | IHM ($\lambda = 0$) | IHM ($\lambda = 0.1$) | Data |
|---|---|---|---|
| 1.0 | 1.0 | 1.0 | IHM |
| | 1.0 | 1.0 | SDM |
| 0.95 | 1.0 | 1.0 | IHM |
| | 0.88 | 0.79 | SDM |
| 0.90 | 0.99 | 1.0 | IHM |
| | 0.58 | 0.66 | SDM |

***Table 4.7:*** *The ratio of number of robust matches over the number of all matches among fingerprints computed with autoencoders trained on SDM with respect to the changes in matching threshold, $\lambda$, and data. The matching trends shows that the sparse AE ($\lambda = 0.3$, trained on SDM) outperforms all other models (including the ones in Table 4.6) on both IHM and SDM data. In other words, even when the threshold is decreased, the ratio of the robust matches stay stable, exhibiting smooth latent space properties. Note that the robust matches are determined based on the tables presented in Figures 4.10 and 4.11.*

| Threshold | SDM ($\lambda = 0$) | SDM ($\lambda = 0.3$) | Data |
|---|---|---|---|
| 1.0 | 1.0 | 1.0 | IHM |
| | 1.0 | 1.0 | SDM |
| 0.95 | 1.0 | 1.0 | IHM |
| | 0.87 | 1.0 | SDM |
| 0.90 | 0.98 | 1.0 | IHM |
| | 0.59 | 1.0 | SDM |

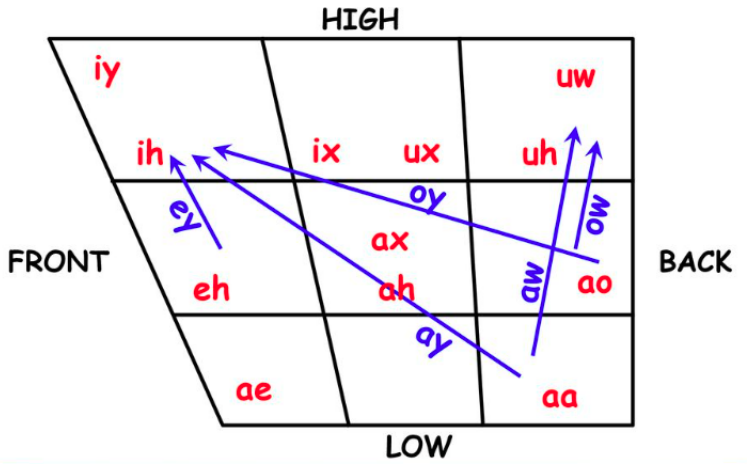



***Figure 4.11:*** *American English Vowel Space [Vendetti (2002b)].The matches between fingerprint vectors mapping to similar phone labels in terms of manner and/or place of articulation are taken as robust match. The details of the analysis whose goal is to investigate whether the representation space (where fingerprint vectors reside) exhibits orthogonalization and/or smooth latent space properties (Section 3.2.2) can be found in Tables 4.6 and 4.7. The manifestation of these properties is important for understanding the representation capacity of the AE encodings.*

Overall datasets and AE models, Sparse AE trained on SDM data (i.e., SDM $\lambda = 0.3$ in Table 4.7) demonstrates the most robust performance, with respect to the changes in the threshold. This supports our standpoint that SDM contains more distortion compared to IHM, and hence applying sparsity on the models, which are trained on SDM (i.e., SDM($\lambda = 0.3$), helps learning robust and meaningful encodings (i.e., embeddings). The matching trends of the fingerprint vectors computed with SDM $\lambda = 0.3$ model (demonstrated in Table 4.7) outperform all other models (including the ones in Table 4.6) on both IHM and SDM data. In other words, even when the threshold is decreased, for sparse AE on distorted data (i.e., SDM $\lambda = 0.3$), the ratio of robust matches (over all matches) can stay stable, exhibiting smooth latent space properties.

## 4.5 Conclusion

In this chapter, we introduced our proposed approach for low-rank and sparse modeling of LF-MMI log-likelihoods by means of undercomplete and sparse autoencoders, respectively.

We inserted an additional autoencoder component between the acoustic model and decoder components in the ASR pipeline (Figure 2.1). Once the autoencoder was trained, we studied the reconstructed log-likelihoods (from the autoencoder output layer) for word recognition. With a strong state-of-the-art LF-MMI baseline system (Section 2.5), the log-likelihoods were challenging to be further improved by means of our proposed sparse modeling approach. Hence, we could only obtain promising improvement in terms of WER (Table 4.2) on far-field speech.

LF-MMI log-likelihoods are already robust, discriminative and task-driven (i.e., mostly containing ASR-relevant components) due to the state-of-the-art acoustic model training. Hence, in the following chapter, we will work on high-resolution MFCC features. Along this line, we anticipate that:

1. Our proposed approach with autoencoders can actually be useful for learning more useful representations out of MFCCs and consequently improve WER.
2. Going overcomplete (on the encoding layer) will be less computationally demanding and greatly expand the modeling space for MFCC features. Provided that MFCCs are lower in dimensionality (i.e., 176 dimensional LF-MMI log-likelihoods versus 40 dimensional MFCCs), they will be given more space to scatter and satisfy smooth latent space properties, even with the same encoding dimension.

The subspace analysis on high-dimensional sparse features demonstrated that sparsity and overcompleteness (Section 3.2.2) especially help the models trained on far-field SDM data for robust speech modeling (Table 4.7). This is mostly due to the SDM's distorted nature compared to close-field IHM. To be more precise, in the subspace analysis (Section 4.4.2), these features are expected to reside in a vector space, given a similarity measure (i.e., cosine similarity) and a matching threshold. And, we studied the impact of sparsity for robust matches among feature vectors. Mainly, the content (based on phone-level and articulatory-level knowledge) of the matches were meaningful (Table 4.7).

For further reading on our findings regarding speech modeling and speech recognition with Sparse AE $\ell_1$ norm penalty, we refer reader to [Kabil and Bourlard (2022c)] and [Kabil and Bourlard (2022b)], respectively.

# 5 Low-Rank and Sparse Modeling of Acoustic Features

In this chapter [1], we apply our proposed approach which makes use of undercomplete autoencoders for low-rank modeling and sparse overcomplete autoencoders for sparse modeling of acoustic features, namely Mel-Frequency Cepstral Coefficients (MFCCs).

In summary, we now insert an additional autoencoder component between the feature extractor and the acoustic model components in the ASR pipeline (Figure 2.1). The autoencoder component is expected to generate new (i.e., reconstructed) and better features so that they can lead to performance improvements when used for training the LF-MMI acoustic model with the baseline acoustic model configuration (Section 2.5). Upon autoencoder training, the reconstructed MFCCs are sent to the LF-MMI acoustic model. Next, once the acoustic model is trained, LF-MMI log-likelihoods are sent to the decoder component for ASR decoding.

The proposed sparse modeling approach (with sparse overcomplete autoencoder) yields improvement on recognizing far-field SDM data. However, the analysis on high-dimensional sparse features (i.e., encodings from sparse overcomplete autoencoder) displays limitations for modeling speech components, presenting less informative overview on speech subspaces compared to its counterpart in Chapter 4.

## 5.1 Introduction

Due to the difficulties for improving WER in the previous chapter, we focus on acoustic features (i.e., input features for the acoustic model), rather than LF-MMI log-likelihoods (i.e., output features from the acoustic model). LF-MMI log-likelihoods are already discriminant and task-driven, due to propagation through several layers in the DNN acoustic model whose training objective is to model the relationship between the audio signal and the phonemes or other linguistic units that constitute speech. Therefore, it was challenging to further improve

[1] This chapter is partially based on the following publications:
Kabil, S.H., and Bourlard, H. (2022). Speech modeling using sparse autoencoders.
Kabil, S.H., and Bourlard, H. (2022). Sparse autoencoders to enhance speech recognition.

these log-likelihoods.

As MFCCs lack propagation across the layers of DNN acoustic model, they are not as robust or task-driven as log-likelihoods. Hence, we hypothesize that autoencoders can be useful for modeling the inner underlying components of MFCCs, even though not all of these components are strictly related to speech recognition and modeling tasks.

In addition, in the case of sparse overcomplete autoencoders, as the ratio between input dimension and encoding layer dimension is higher, we present higher modeling capacity (i.e., larger representation space) for the underlying components to scatter, in a less computationally demanding manner (compared to the models in Chapter 4). We expect this to be especially beneficial for easier explanation of the analysis on high-dimensional sparse encodings.

Accordingly, since our primary focus in this thesis is sparse modeling (of speech data) in a generic and unsupervised manner, by means of sparse autoencoders, we present the low rank modeling experiments with undercomplete autoencoders, only for the sake of completeness.

## 5.2 Our Approach

We insert an Autoencoder (AE) between the feature extractor and the acoustic model components in the ASR pipeline (Figure 2.1). The low-rank and sparse modeling of MFCCs are handled by undercomplete (Figure 5.1) and sparse overcomplete (Figure 5.3) autoencoders, respectively.

After unsupervised AE training, the reconstructed MFCCs are sent to the LF-MMI acoustic model (Section 2.2) for training and then to the Kaldi decoder for ASR decoding, as illustrated with blue arrow in Figures 5.1 and 5.3. The word recognition performances in WER are presented in Tables 5.1 and 5.2.

In addition, (again, once AE training is finished), to assess the representation capacity of AE encodings (i.e., hidden unit activations, embeddings), a simple DNN phone classifier is trained on them (as illustrated with red arrow in Figures 5.1 and 5.3). The frame-level phone accuracies are reported in Table 5.3.

### 5.2.1 Undercomplete Autoencoders to Enhance MFCCs

In Figure 5.1, $\mathbf{x}$ represents 40 dimensional high-resolution MFCC feature vector and $\tilde{\mathbf{x}}$ denotes its reconstructed version. In undercomplete AE, the encoding $\mathbf{z}$ is lower in dimension than $\mathbf{x}$.

To determine the optimal dimension for $\mathbf{z}$ in autoencoders on close-field IHM and far-field SDM speech, we examine the elbow region on the development sets (Figure 5.2). We set the bottleneck dimension for $\mathbf{z}$ as 30 for both undercomplete AEs trained on IHM and SDM.

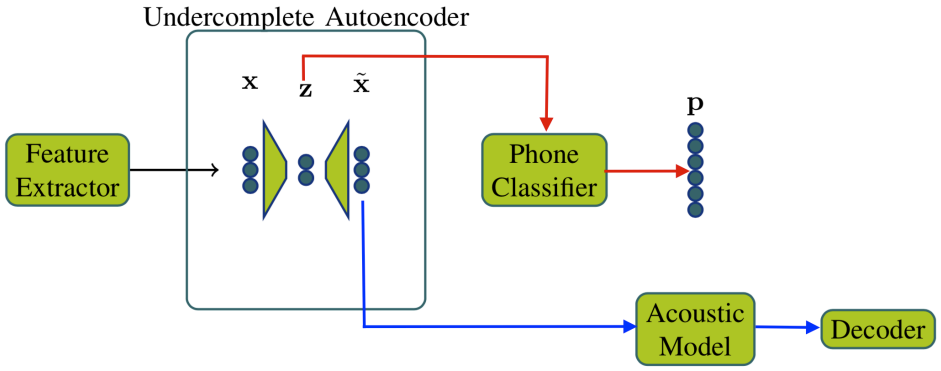



***Figure 5.1:*** *Low-rank modeling of MFCCs by means of undercomplete AE. The reconstructed MFCCs are sent to the LF-MMI acoustic model and then to the decoder for word recognition. In addition, AE encodings are sent to a simple DNN phone classifier for frame-level phone classification.*

The autoencoders are implemented in Pytorch [Paszke et al. (2017)]. Note that all other components are implemented in Kaldi [Povey et al. (2011)] speech processing toolkit. For reading and writing Kaldi data format (e.g., ark files), we use a pure Python module called *kaldiio* [2].

The MFCCs are preprocessed by mean-normalization via Kaldi's *apply-cmvn* command. We observe that this normalization scheme is helpful for faster convergence with autoencoders. All autoencoders are trained for (maximum of) 100 epochs with Adam optimizer and scheduler for early stopping with patience of 5 epochs.

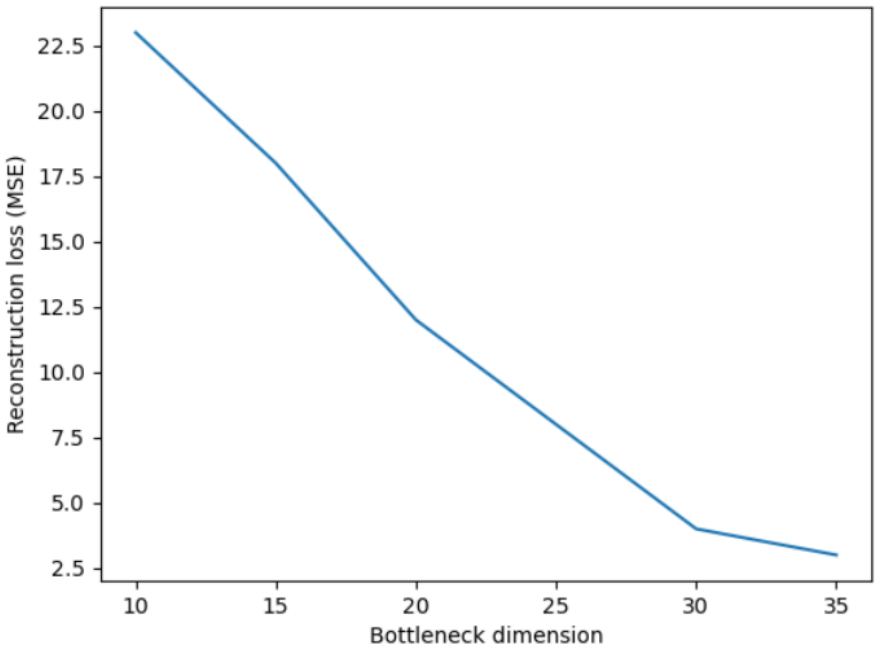
After AE training, the reconstructed MFCCs are fed to the LF-MMI acoustic model for training. After acoustic model training is done, LF-MMI log-likelihoods (i.e., LF-MMI acoustic model outputs) are sent to the Kaldi decoder. This time, unlike Chapter 4, Kaldi's *nnet3-latgen-faster* command can be used for decoding graph generation.

In other words, instead of passing the MFCCs (from the feature extractor) directly to the LF-MMI acoustic model (indicated as baseline in Tables 5.1 and 5.2), we send their reconstruction from undercomplete AE to the acoustic model (indicated as Undercomplete AE in Table 5.1). We observe that low-rank modeling setup results in degradation for SDM.

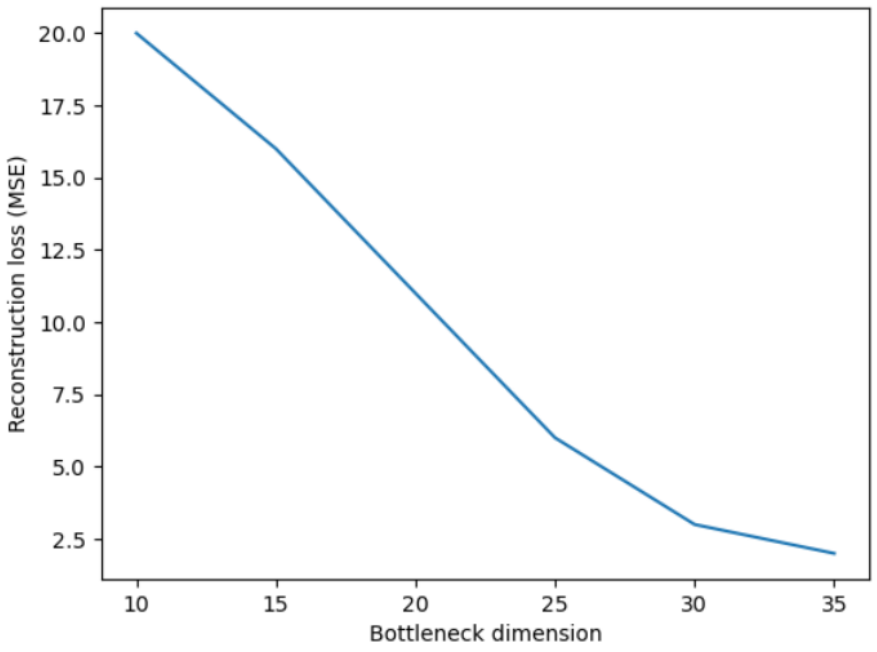
***Table 5.1:*** *The recognition performance (in WER%) for LF-MMI system and the proposed approach on IHM and SDM evaluation sets. Low-rank modeling leads to degradation in WER for far-field SDM data.*

| Architecture | $IHM_{WER}$ | $SDM_{WER}$ |
|---|---|---|
| Baseline | 19.2 | 41.1 |
| Undercomplete AE | 19.3 | 42.3 |

[2] https://github.com/nttcslab-sp/kaldiio




**(a)** IHM




**(b)** SDM

***Figure 5.2:*** *Optimal bottleneck dimension for undercomplete AEs are determined by observing the elbow region (L-curve) on development set. The bottleneck dimension is set 30 for both undercomplete AEs trained on IHM and SDM accordingly.*

### 5.2.2 Sparse Overcomplete Autoencoders to Enhance MFCCs

As stated in Chapter 4, our goal is to achieve sparse modeling of speech data in a generic and unsupervised manner by means of sparse autoencoders. In [Dighe (2019)], sparse modeling approach is shown to be better at modeling speech data as union of low-dimensional subspaces, especially for far-field SDM data.

Revisiting the duality between dictionary learning [Mairal et al. (2009)] and sparse autoencoders, we implement shallow overcomplete sparse autoencoder with $\ell_1$ norm constraint on hidden unit activations with loss function formulated as (3.6). In addition, to ensure full compliance with dictionary learning (3.7), the encoder and decoder weights are tied (i.e., transpose of each other), the encoder weights are normalized, and no bias parameter is used.

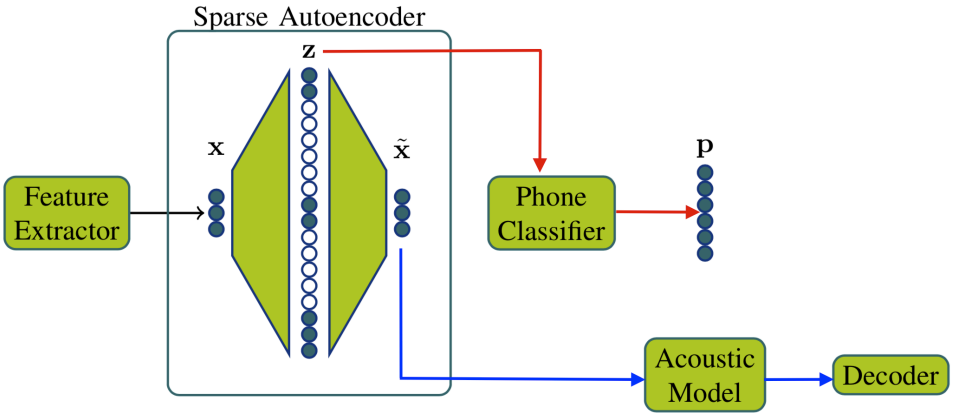



***Figure 5.3:*** *Sparse modeling of MFCCs by means of sparse overcomplete AE. The reconstructed MFCCs are sent to the acoustic model and then to the decoder. The recognition performances are presented in Table 5.2. Furthermore, once AE training is finished, simple DNN based phone classifier is trained on AE encodings to assess their representation power. The frame-level phone accuracies are reported in Table 5.3.*

Thus, we transform the dictionary learning problem to a representation learning problem by means of sparse autoencoders. When sparse AE is trained to solve (3.6), forward pass can be viewed as the sparse coding step in dictionary learning, since we obtain high-dimensional sparse representation (i.e., encoding) $\mathbf{z}$ on the hidden layer. Similarly, the backward pass is analogous to the dictionary update step in dictionary learning, as decoder weights (akin to the overcomplete dictionary in dictionary learning setup) are updated based on the distance between the original input $\mathbf{x}$ and reconstructed input $\tilde{\mathbf{x}}$. In Figure 5.3, $\mathbf{x}$ represents the 40 dimensional MFCC vector, and $\mathbf{z}$ stands for the overcomplete (i.e., higher-dimensional) encoding vector with dimensionality of 1760. To be consistent with the models in Chapter 4, we keep the AE encoding dimensionality as 1760, which is $176 \times 10$, where 176 is the number of senones in our Kaldi setup for AMI database and the scalar 10 is empirically decided so that MFCCs can scatter in the encoding space.

To determine the optimal $\lambda$ in (3.6) for weighting the sparsity penalty term, we follow the L-curve on development sets, formed by sparsity penalty (i.e., unweighted $\ell_1$ sum) versus reconstruction loss (i.e., MSE), as illustrated in Figure 5.4. The optimal $\lambda$ coefficients for IHM and SDM are found as 0.0001 and 0.0005, respectively.

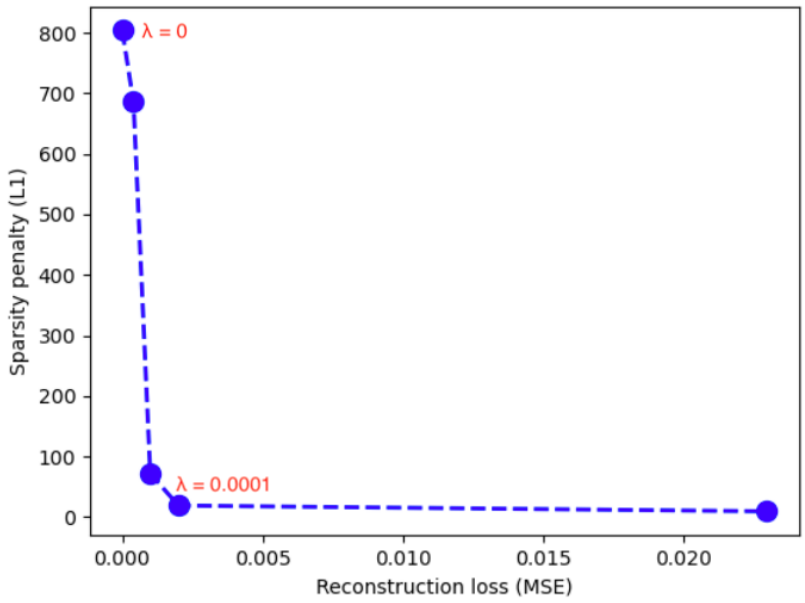
In full accordance with the implementation details in Section 5.2.1, the autoencoders are implemented in Pytorch [Paszke et al. (2017)], following the same preprocessing and training procedures.

In other words, instead of passing the MFCCs from the feature extractor directly to the LF-MMI acoustic model (indicated as baseline in Tables 5.1 and 5.2), we send their reconstruction from shallow overcomplete sparse AE to the acoustic model (indicated as Sparse AE in Table 5.2). As anticipated, the sparse modeling approach by means of sparse AE works better for far-field SDM data.

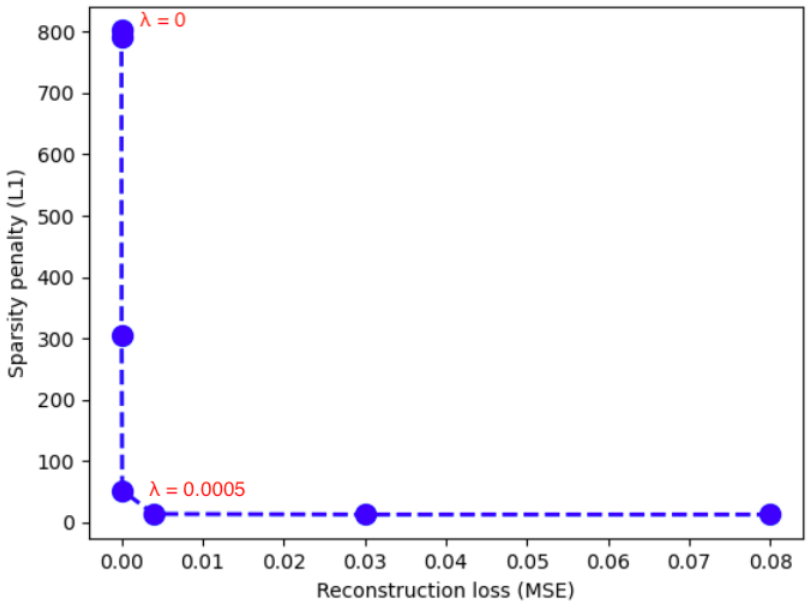
For the sake of completeness, we also report the results for vanilla ($\lambda = 0$) overcomplete AE. In line with findings in [Dighe (2019)], sparse modeling approach works better for far-field SDM data (i.e., improvement on WER with overcompleteness and sparsity) than close-field IHM, as shown in Table 5.2.

***Table 5.2:*** *The recognition performance (in WER%) for baseline LF-MMI system and the proposed approach on IHM and SDM evaluation sets. To present the full picture, the results for Vanilla ($\lambda = 0$) overcomplete AE are also reported. As anticipated, the proposed sparse modeling approach (denoted as Sparse AE) works better for far-field SDM data.*

| Architecture | $IHM_{WER}$ | $SDM_{WER}$ |
|---|---|---|
| Baseline | 19.2 | 41.1 |
| Overcomplete AE ($\lambda = 0$) | 19.4 | 40.6 |
| Sparse AE | 19.7 | 40.5 |




**(a)** IHM




**(b)** SDM

***Figure 5.4:*** *Optimal sparsity penalty coefficients for sparse overcomplete AEs are determined by observing the elbow region (L-curve) on development set, formed by sparsity penalty (i.e., unweighted $\ell_1$ sum) versus reconstruction loss (i.e., MSE). The optimal $\lambda$ coefficients for (a)IHM and (b)SDM are 0.0001 and 0.0005, respectively.*

## 5.3 Frame-level Phone Accuracy

Apart from exploring the use of reconstructed MFCCs for speech recognition (Section 5.2), we also examine the practical application of AE encodings in the same context. Following the same reasoning stated in Section 4.3, we train simple classifier neural networks on these encodings for frame-level phone classification task.

In full accordance with the implementation details in Section 4.3, the classifiers are implemented using Pytorch, and trained with cross-entropy (CE) criterion. We use *ali-to-phones* command with *–per-frame=true* flag in Kaldi to extract the forced (frame-level) phone alignments. These alignments are then used as labels for the classifier.

***Table 5.3:*** *The frame-level phone classification accuracies (in %) on IHM and SDM evaluation sets. The encodings from sparse overcomplete AE obtain better performance for far-field SDM than close-field IHM. The fact that MFCCs are not as task-driven as log-likelihoods and the DNN phone classifiers are indeed simple leads to the performance gap presented in Tables 4.3 and 5.3.*

| Architecture | $IHM_{accuracy}$ | $SDM_{accuracy}$ |
|---|---|---|
| Baseline | 39.2 | 33.7 |
| Undercomplete AE | 39.0 | 33.0 |
| Overcomplete AE ($\lambda = 0$) | 38.6 | 33.3 |
| Sparse AE | 38.0 | 33.5 |

We observe that encodings from sparse overcomplete AE provide better representation power on far-field SDM (i.e., 0.2% below the performance of the classifier trained on MFCCs), compared to close-field IHM (i.e., more than 1% under the performance of the phone classifier trained on MFCCs). This performance gap might be resulted from the fact that MFCCs are not task-driven (i.e., also contain components that are not necessarily helpful for recognition tasks) and encodings are learned based on reconstruction objective. It is important to note that better reconstruction loss does not necessarily mean learning better and/or meaningful features.

Finally, as for the performance gap between Tables 4.3 and 5.3, it results from the fact that MFCCs are less task-driven and thus less discriminative (compared to senone log-likelihoods), while phone classifiers are indeed really simple to disentangle the underlying components of MFCCs in a discriminative manner. In the next section, we analyze whether these encodings are indeed sparse and meaningful.

## 5.4 Analysis on High-dimensional Sparse Features

As shown in Table 5.3, we observe that high-dimensional encodings from sparse AE obtain promising performance for frame-level phone classification, especially for far-field SDM data, which is known to have more distortions than close-field IHM.

In this section, we first examine if the $\ell_1$ norm sparsity constraint (imposed on Sparse AE) behaves as expected and hence guides the AE to produce high-dimensional sparse encodings. Along this line, to evaluate the sparsity of hidden unit activations (i.e., encodings), we utilize $\ell_0$, $\ell_0^\epsilon$ and Hoyer measure (introduced in Section 3.2.5).

Then, in Section 5.4.2, we investigate if the encodings space (i.e., representation space) exhibits orthogonalization or smooth latent space properties (Section 3.2.2) when cosine similarity is used as similarity measure.

### 5.4.1 Sparsity of Activations

As previously explained in Section 3.2.1, the definition for sparseness can differ. Here, we examine if the hidden unit activations (i.e., AE encodings) are indeed sparse, in the sense that **(1)** they contain hard-zeros, or **(2)** only a few dimensions of the activation vector (i.e., encoding vector) carry most of the information content (while none of the dimensions are not necessarily hard-zero).

Regarding hard-zeros **(1)**, we utilize $\ell_0$ and $\ell_0^\epsilon$ measures (Section 3.2.5). Based on $\ell_0$ measure, we observe that we do not obtain hard-zero activations (shown in Tables 5.4 and 5.5). On the other hand, $\ell_0^\epsilon$ with $\epsilon$ set according to the value range on development sets, we see that sparsity constraint helps to pull the activations towards zero, although fails to produce hard-zero activations as expected (illustrated in Tables 5.4 and 5.5, second column).

***Table 5.4:*** *The sparsity of the hidden unit activations based on the complements of $\ell_0$ and $\ell_0^\epsilon$ measures for (a randomly chosen) subset of utterances from IHM development set. For simplicity, instead of total number of zero entries, we provide their percentage over all entries in the activation vectors for (randomly chosen) subset of utterances. We observe that the sparsity constraint ($\lambda > 0$, optimally being 0.0001 for IHM) pulls hidden unit activations towards zero, although fails to generate hard-zeros.*

| Architecture | $(\ell_0)^c$ (in %) | $(\ell_0^\epsilon)^c$ (in %) |
|---|---|---|
| IHM ($\lambda = 0$) | 0 | 5 |
| IHM ($\lambda = 0.0001$) | 0 | 9 |

***Table 5.5:*** *The sparsity of the hidden unit activations based on the complements of $\ell_0$ and $\ell_0^\epsilon$ measures for (a randomly chosen) subset of utterances from SDM development set. For simplicity, instead of total number of zero entries, we provide their percentage over all entries in the activation vectors for (randomly chosen) subset of utterances. We observe that the sparsity constraint ($\lambda > 0$, optimally being 0.0005 for SDM) pulls hidden unit activations towards zero, although fails to generate hard-zeros.*

| Architecture | $(\ell_0)^c$ (in %) | $(\ell_0^\epsilon)^c$ (in %) |
|---|---|---|
| SDM ($\lambda = 0$) | 0 | 2 |
| SDM ($\lambda = 0.0005$) | 0 | 6 |

Regarding information carried by active units **(2)**, we utilize Hoyer measure [Hoyer (2004)], where $v$ with dimension $n$ in (3.10) stand for the hidden unit activation vector $z$ in Figure 5.3, with dimension 1760, respectively. It is important to note that sparseness based on Hoyer measure does not necessarily mean that the entries (i.e., dimensions, coefficients) of the activation vector (i.e., encoding vector, representation vector) are hard-zero. Hoyer measure generates scores in $[0,1]$ range where 0 denotes the least sparse case (i.e., uniform distribution of information among entries of the hidden unit activation vector) and 1 denotes the most sparse case (i.e., spiky distribution, almost all information is carried by only one coefficient).

Focusing on the *same* randomly chosen subset of utterances from development sets (previously used in Chapter 4.4), we calculate the sparseness of the hidden unit activations, with Hoyer measure. Figures 5.5 and 5.6 for sparseness measurement of the hidden unit activation vectors for (a) Vanilla $\lambda = 0$ overcomplete and (b) Sparse overcomplete AE on IHM and SDM, respectively. $x$ axis shows the Hoyer score, $y$ stands for the number of activation vectors. It is important to note that each (input) MFCC vector (i.e., frame) has a corresponding activation vector revealed on the AE's encoding layer. Hence, histograms clearly depict the alteration in the characteristics of the activations (i.e., encodings) in the presence of sparsity.

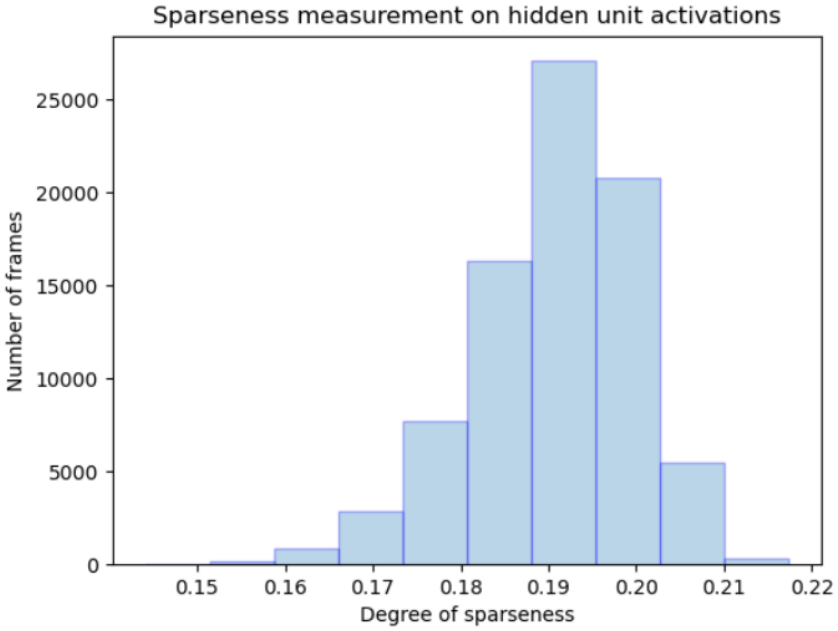
Table 5.4, Figure 5.5 (for IHM) and Table 5.5, Figure 5.6 (for SDM) are computed on the *same* (randomly chosen) subset of utterances from development sets. Therefore, they present a complementary view for the analysis of the sparsity of activations. In conclusion, we observe that $\ell_1$ norm sparsity penalty (on hidden units) guides the AE to force its hidden units to produce close-to-zero activations. In addition, while the majority of the hidden units produce close-to-zero activations, a small number of hidden units produce the majority of the activation (i.e., information, energy), forming the rare spikes among many close-to-zero entries in the activation vector.

In the rest of the analysis, we examine if these spiky hidden unit activations actually represent meaningful patterns, modeling senone (or any other speech unit's) subspaces.

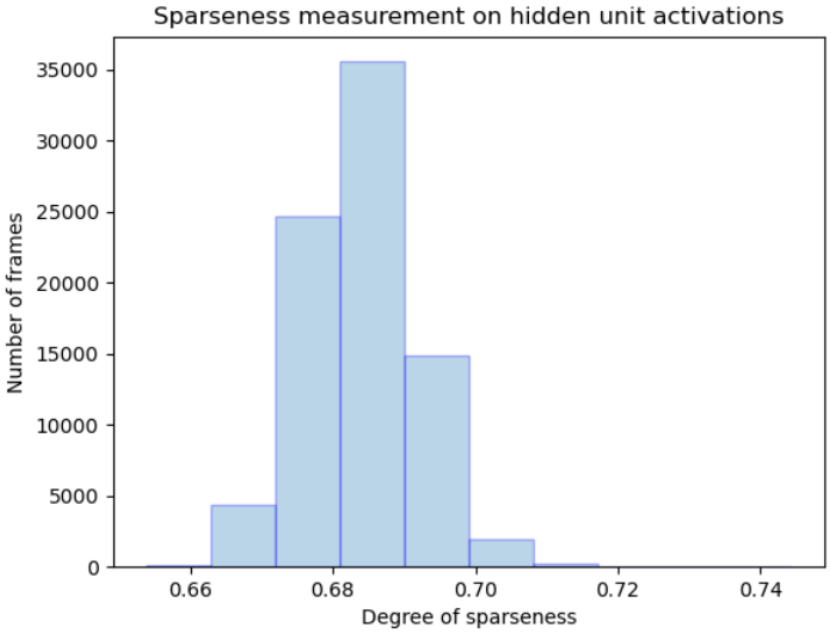
### 5.4.2 Subspace Analysis

With our proposed approach for sparse modeling, we aim for individual subspaces to lie in the common high dimensional representation space in a scattered manner (as shown in Figure 4.1). Our hypothesis is that this behavior can induce the separability of senones (and even phones).

Based on the findings in Section 5.4.1, we anticipate the hidden unit activation vectors for a particular senone to have a distinctive pattern and to share a few common features with other senones that have similar phonetic and articulatory characteristics. Note that senones are formed by clustering the phones based on the decision tree questions regarding contextual and phonetic properties. That is why, later in our analysis here, we take the phones as the disentangled form of senones.

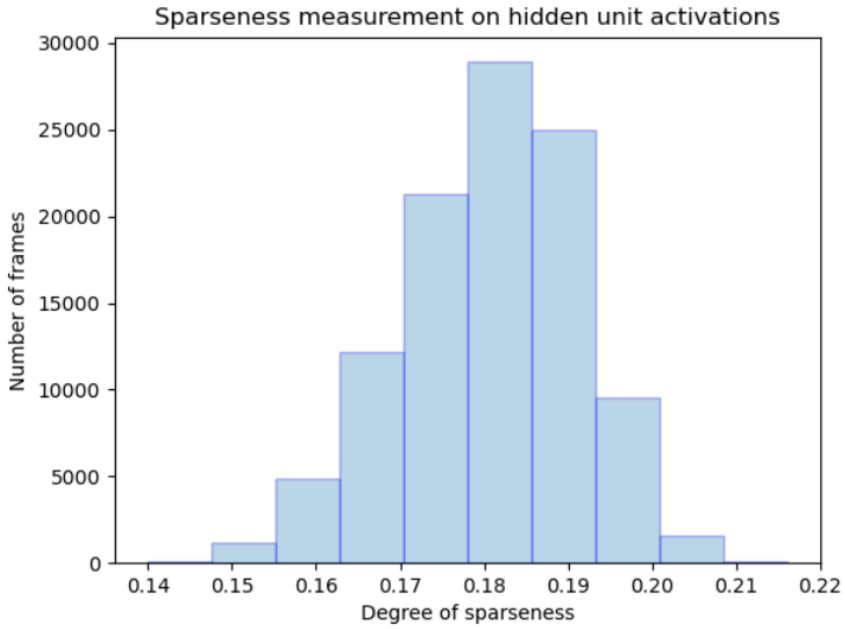



**(a)** IHM $\lambda$=0

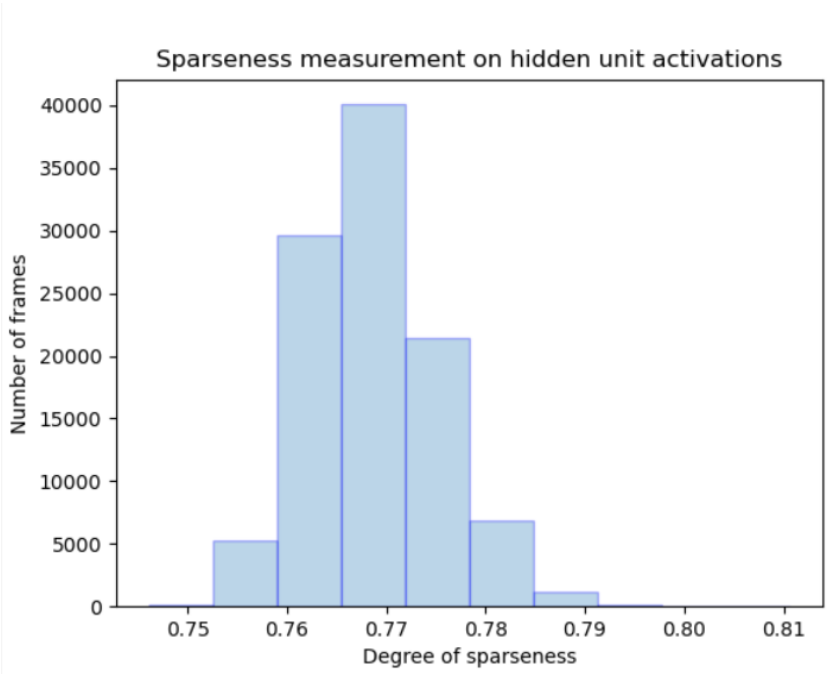



**(b)** IHM $\lambda$=0.0001

***Figure 5.5:*** *Sparseness measurement on the hidden unit activations of randomly chosen subset of utterances from IHM development set. Enforcing sparsity $\lambda > 0$ is observed to change the characteristics of the activations. For Vanilla $\lambda = 0$ overcomplete AE (a), majority of the activation vectors have 0.19; whereas, for sparse overcomplete AE (b), majority of the activation vectors have 0.69 as Hoyer sparseness score. On the scale of $[0, 1]$, 0 denotes the least sparse, 1 denotes the most sparse case.*




**(a)** SDM $\lambda$=0




**(b)** SDM $\lambda$=0.0005

***Figure 5.6:*** *Sparseness measurement on the hidden unit activations of randomly chosen subset of utterances from SDM development set. Enforcing sparsity $\lambda > 0$ is observed to change the characteristics of the activations. For Vanilla $\lambda = 0$ overcomplete AE (a), majority of the activation vectors have 0.18; whereas, for sparse overcomplete AE (b), majority of the activation vectors have 0.77 as Hoyer sparseness score. On the scale of $[0,1]$, 0 denotes the least sparse, 1 denotes the most sparse case.*

Hence, focusing on the *same* randomly chosen subset of utterances in Section 5.4.1 (with 419 utterances and 112386 frames), we average all activation vectors belonging to a particular senone id (i.e., class) and set the resulting vector as *fingerprint* for that particular senone. Similar to Section 4.4.2, inspired from sparse distributed memory [Kanerva (1988)], we assume all fingerprint vectors lie in a vector space (i.e., representation space) with cosine similarity (5.1) as similarity metric. Shortly, we hypothesize that we observe smooth latent space properties (Section 3.2.2) to emerge on the vector space where the fingerprint vectors reside.

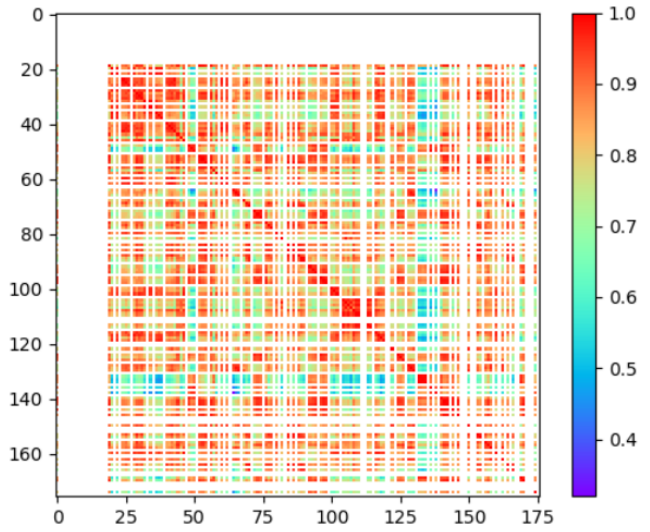
Figures 5.7 and 5.8 illustrate the similarities between the observed senone classes in the subset among overall 176 senones in the setup. Similarity scores are computed via cosine similarity metric (5.1). Cosine similarity produces scores $[-1,1]$, 1 denotes the highest similarity (shown in red in figures).

$$\text{similarity } = \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}, \tag{5.1}$$

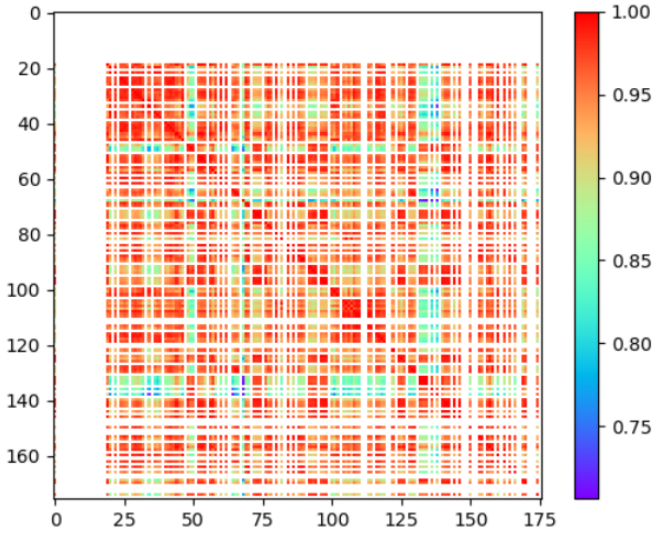
We observe that AE encodings are indeed sparse, especially with respect to Hoyer measure. Compared to Tables 4.4 and 4.5, $\ell_0^\epsilon$ does not produce high percentages of zero activations. Hence, the product of fingerprint vectors (during cosine similarity computation) does not converge to zero at all, which is reflected on the colormaps in Figures 5.7 and 5.8 and their legends (i.e., value range of the colormap legends). The colormaps are dominated by red color; hence, they indicate high similarity and provide little information, compared to Figures 4.8 and 4.9. This is probably due to MFCCs being less task-driven (i.e., more entangled) and less robust, compared to senone log-likelihoods (from Chapter 4). We therefore hypothesize that this will impact the total number of matches and the ratio of robust matches badly for phone-level subspace analysis, compared to the one presented in Chapter 4.

Nevertheless, based on our findings with colormaps, we extend our subspace analysis with fingerprint vectors. Thanks to Kaldi's *show-transitions*, *ali-to-pdf* and *ali-to-phones* commands, we manage to generate mappings between the senones (i.e., pdfs in Kaldi terminology) and phones with positional encodings. We anticipate that fingerprint vectors might be more informative on the phone-level patterns which are in more human interpretable form.

Using Figures 5.9 and 5.10, we take the matches between fingerprint vectors, mapping to similar phone labels in terms of manner and/or place of articulation, as *robust* match. For instance, given cosine similarity as metric and a matching threshold, the match between the fingerprints for /s/(fricative, alveolar) and /z/(fricative, alveolar) is robust. On the other hand, the match between the fingerprints for /s/ and /m/(nasal, bilabial) is a false match (i.e., not robust match); hence, not desired, especially when the matching threshold is decreased.

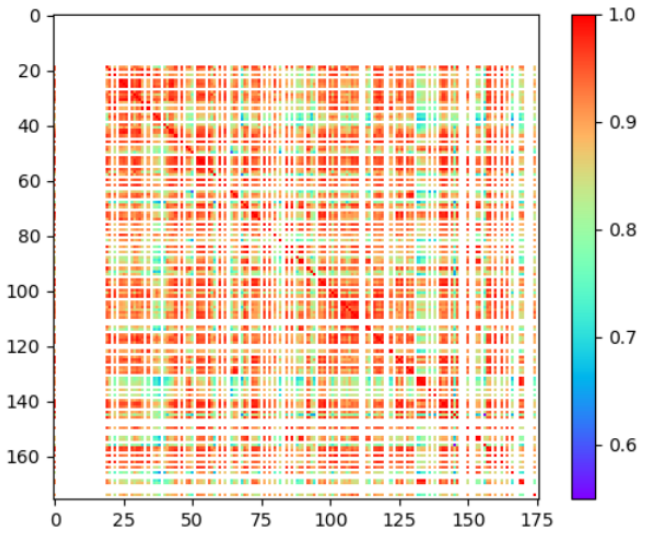



**(a)** IHM $\lambda$=0

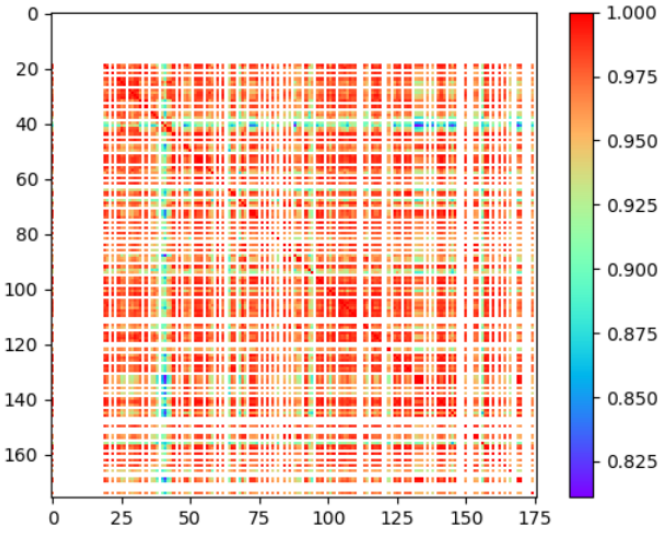



**(b)** IHM $\lambda$=0.0001

***Figure 5.7:*** *Colormaps for visualizing the similarity analysis between the senone-specific fingerprint vectors. Note that not all 176 senones (in the setup) occur in the subset of utterances, which is reflected as white regions in the colormaps above. The similarities between the (fingerprints of) observed senones are computed with cosine similarity. We observe that in the colormaps are dominated by red color; hence, indicate high similarity which can also be traced in the shrinkage on the value range of the colormap legends (i.e.,* $[0.4, 1]$ *for* $\lambda = 0$*, whereas* $[0.75, 1]$ *for* $\lambda = 0.0001$*). Hence, compared to Figures 4.8 and 4.9, these colormaps provide little information. This is probably due to MFCCs being less task-driven (i.e., more entangled) and less robust, compared to senone log-likelihoods..*




**(a)** SDM $\lambda$=0




**(b)** SDM $\lambda$=0.0005

***Figure 5.8:*** *Colormaps for visualizing the similarity analysis between the senone-specific fingerprint vectors. Note that not all 176 senones (in the setup) occur in the subset of utterances, which is reflected as white regions in the colormaps above. The similarities between the (fingerprints of) observed senones are computed with cosine similarity. We observe that in the colormaps are dominated by red color; hence, indicate high similarity (even in the presence of sparsity) which can also be traced in the shrinkage on the value range of the colormap legends (i.e.,* $[0.6, 1]$ *for* $\lambda = 0$*, whereas* $[0.83, 1]$ *for* $\lambda = 0.0005$*). Hence, compared to Figures 4.8 and 4.9, these colormaps provide little information. This is probably due to MFCCs being less task-driven (i.e., more entangled) and less robust, compared to senone log-likelihoods.*

We examine the ratio of number of *robust* matches over total matches with respect to the changes in the matching threshold, $\lambda$ and data (i.e., on which fingerprint vectors are computed). As shown in Tables 5.6 and 5.7, with the decrease in threshold, the number of total matches increases faster than the number of robust matches (i.e., the ratio decreases). The increase in $\lambda$ degrades the ratio of robust matches over all matches.

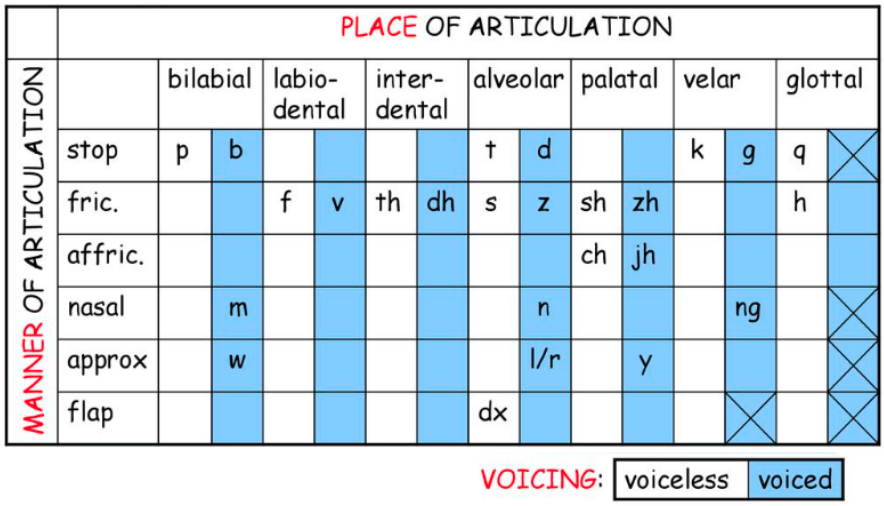



***Figure 5.9:*** *Articulatory parameters for English consonants in ARPAbet [Vendetti (2002a)]. The matches between fingerprint vectors mapping to similar phone labels in terms of manner and/or place of articulation are taken as robust match. The details of the analysis whose goal is to investigate whether the representation space (where fingerprint vectors reside) exhibits orthogonalization and/or smooth latent space properties (Section 3.2.2) can be found in Tables 5.6 and 5.7. The manifestation of these properties is important for understanding the representation capacity of the AE encodings.*

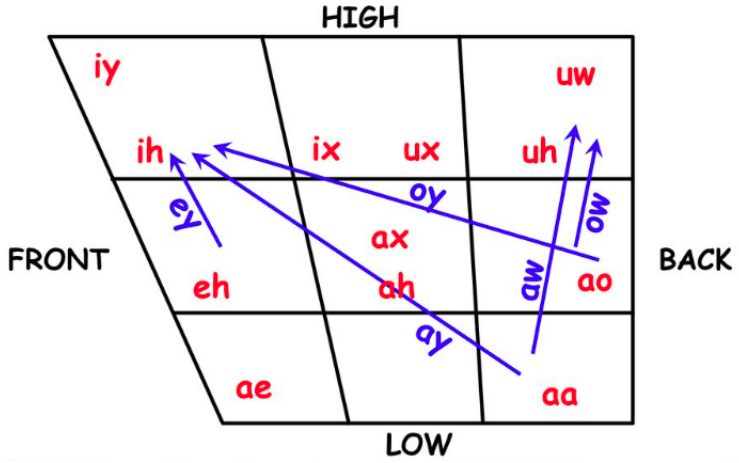



***Figure 5.10:*** *American English Vowel Space [Vendetti (2002b)].The matches between fingerprint vectors mapping to similar phone labels in terms of manner and/or place of articulation are taken as robust match. The details of the analysis whose goal is to investigate whether the representation space (where fingerprint vectors reside) exhibits orthogonalization and/or smooth latent space properties (Section 3.2.2) can be found in Tables 5.6 and 5.7. The manifestation of these properties is important for understanding the representation capacity of the AE encodings.*

***Table 5.6:*** *The ratio of number of robust matches over the total number of matches (in [0-1] scale) with respect to the changes in the matching threshold, $\lambda$, and data. With decrease in matching threshold, we expect this ratio to get closer to 0 (as the total number of matches is expected to increase swiftly), unless the model has robust modeling capacity to stabilize the ratio of robust matches over all matches. The increase on $\lambda$ degrades the performance for IHM models, especially in the case of SDM data. Note that the robust matches are determined based on the tables presented in Figure 5.9 and Figure 5.10.*

| Threshold | IHM ($\lambda = 0$) | IHM ($\lambda = 0.0001$) | Data |
|---|---|---|---|
| 1.0 | 1.0 | 1.0 | IHM |
| | 1.0 | 1.0 | SDM |
| 0.99 | 1.0 | 0.91 | IHM |
| | 1.0 | 0.36 | SDM |
| 0.98 | 1.0 | 0.45 | IHM |
| | 1.0 | 0.44 | SDM |

***Table 5.7:*** *The ratio of number of robust matches over the total number of matches (in [0-1] scale) with respect to the changes in the matching threshold, $\lambda$, and data. With decrease in matching threshold, we expect this ratio to get closer to 0 (as the total number of matches is expected to increase swiftly), unless the model has robust modeling capacity to stabilize the ratio of robust matches over all matches. In this manner, SDM sparse AE ($\lambda = 0.0005$, trained on SDM) outperforms IHM sparse AE ($\lambda = 0.0001$, trained on IHM) in Table 5.6. In other words, even when the threshold is decreased, the ratio of the robust matches stay stable, exhibiting smooth latent space properties. However, the increase on $\lambda$ degrades the robust modeling capacity of the SDM sparse AE. This implies that SDM data might not be as distorted as we anticipate. Note that the robust matches are determined based on the tables presented in Figures 5.9 and 5.10.*

| Threshold | SDM ($\lambda = 0$) | SDM ($\lambda = 0.0005$) | Data |
|---|---|---|---|
| 1.0 | 1.0 | 1.0 | IHM |
| | 1.0 | 1.0 | SDM |
| 0.99 | 1.0 | 1.0 | IHM |
| | 1.0 | 0.86 | SDM |
| 0.98 | 1.0 | 0.91 | IHM |
| | 1.0 | 0.57 | SDM |

Overall datasets and Sparse AE models, Sparse AE trained on SDM data (i.e., SDM $\lambda = 0.0005$ in Table 5.7) demonstrates the most robust performance, with respect to the changes in the threshold. We reach the similar conclusion in Chapter 4, even though, here the decrease in the matching threshold results in higher number of total matches and more false positives (i.e., non-robust matches), mainly due to the lack of close-to-zero or hard-zero activation values in AE encodings.

## 5.5 Conclusion

In this chapter, we applied our proposed approach for low-rank and sparse modeling of MFCCs by means of undercomplete and sparse autoencoders, respectively.

More specifically, we inserted an additional autoencoder component between the feature extractor and the acoustic model components in the ASR pipeline (Figure 2.1). Upon autoencoder training, we sent the reconstructed MFCCs to the acoustic model. Followingly, once the acoustic model was trained, we passed the LF-MMI log-likelihoods to the decoder for ASR decoding.

LF-MMI log-likelihoods are already robust and discriminant (i.e., task-driven, covering mostly ASR-relevant components) due to the state-of-the-art LF-MMI acoustic model training. In Chapter 4, we observed that this phenomena constitutes a challenge to further improve the log-likelihoods and consequently the word recognition performance. In this chapter, we worked on the input acoustic features for the LF-MMI acoustic model component. MFCCs are known to be less robust and can contain components that are not necessarily relevant or

discriminant for speech recognition (i.e., MFCCs are less task-driven).

We expected the autoencoder component to generate new (i.e., reconstructed) and easier-to-discriminate acoustic features so that they can lead to performance improvements when used for training the LF-MMI acoustic model with the baseline acoustic model configuration (Section 2.5). With our experiments and analysis on high-dimensional sparse speech features,

1. We obtained performance improvement on far-field via proposed sparse modeling approach with sparse overcomplete autoencoders (Table 5.2).
2. We observed limitations in speech in subspace analysis. Compared to their counterparts learned upon LF-MMI log-likelihoods in Chapter 4.4, these high-dimensional sparse speech encodings were really sensitive to the changes in the matching threshold and were not withstanding to maintain robust matches among each other. This is mostly due to the nature of MFCC features.

Based on our findings in this chapter, in upcoming Chapter 6, we will:

1. Continue working on MFCC features for training autoencoders.
2. Explore implicitly constrained sparse autoencoders with internal sparsity mechanism to learn encodings with hard-zeros, primarily for the sake of interpretability of the subspace analysis.

# 6 Implicitly Constrained Sparse Autoencoders

In the previous chapter (Chapter 5), we obtained improvements in WER, which encouraged us to use MFCCs as feature set in the rest of the thesis. The MFCCs, however, made it difficult (compared to Chapter 4) for us to detect patterns in the senone (and even phone) subspaces, since they are less task-driven, more entangled, and harder for shallow overcomplete sparse autoencoders with $\ell_1$ norm penalty to produce encodings close to zero.

Hence, in this chapter, we consider sparse autoencoders with internal sparsity mechanisms, mainly k-Sparse autoencoders [Makhzani and Frey (2013)], and Winner-Take-All autoencoders (WTA AE) [Makhzani and Frey (2015)]. As a result of their biologically inspired internal sparsity mechanisms (Section 3.2.3), we anticipate that these models promote hard-zero embeddings in the encoding layer, which is important for interpreting the subspace analysis with cosine similarity.

We observe that WTA AE trained on SDM data yields the most robust performance in speech modeling, with slight increase in false matches, even in the presence of additive noise. However, we do not obtain improvement on WERs, probably due to the distortion in the value range of the reconstructed MFCCs, because of the internal sparsity mechanism (i.e., top-k selection) on the hidden layer activations.

## 6.1 Introduction

In Chapter 4 and Chapter 5, we use sparse autoencoders with $\ell_1$ norm sparsity penalty on hidden unit activations. This configuration is likely to be the most intuitive (among the models introduced under "sparse autoencoder" name, as stated in Section 3.2), tracing back to the sparse coding studies [Olshausen and Field (1997)] and dictionary learning [Candès and Wakin (2008)].

Our findings show that the sparse autoencoders with $\ell_1$ norm penalty do not produce activations with hard zeros. This is the expected behavior, as $\ell_1$ norm penalty only pulls the activations towards zero. Activations with hard zeros are expected to be more useful for steady

analysis of the representation space with a determined similarity measure (e.g., cosine similarity). In addition, we observe that MFCCs constitute a more convenient feature set when WER is the concern. Therefore, in this chapter, we further explore different sparse autoencoder configurations with MFCCs as input features for learning meaningful and generalizable speech representations.

As previously explained in Section 3.2, there exist many different autoencoder configurations proposed under the name "sparse autoencoder" in literature. Given our objective and findings so far, in this chapter, we focus on sparse autoencoders with internal pruning mechanism, such as k-Sparse Autoencoder (k-Sparse AE) [Makhzani and Frey (2013)], Winner-Take-All Autoencoder (WTA AE) [Makhzani and Frey (2015)].

k-Sparse AE (Section 6.2) is implicitly sparse, capable of producing hard zero activations, thanks to its internal top-k selection (i.e., pruning) mechanism. For every data sample, this model keeps only the highest k activations while setting the rest of the activations to zero. Therefore, model performance is largely dependent on the hyper-parameter $k$. k-Sparse AE boosts population sparseness (Section 3.2.3), just like Sparse AE with $\ell_1$ norm penalty. Therefore, it is likely to struggle with dead unit problem (Section 3.2.3), which is one of the main obstacles for learning meaningful representations (e.g., sparse distributed representations).

Winner-Take-All AE (Section 6.3) approaches the dead unit problem with the insights from competitive learning. With winner-take-all concept [Srivastava et al. (2013)] in this model, hidden units are encouraged to compete with each other for activation; hence, be more selective for being responsive (i.e., active) for the input data samples. Its internal pruning mechanism guides the hidden units not to be constantly active or constantly inactive (i.e., dead, zero activation) during their lifetime (i.e., during training). Therefore, this model proposes a solution for the dead unit problem by promoting life-time sparseness (Section 3.2.3).

## 6.2 k-Sparse Autoencoders

k-Sparse Autoencoder (k-Sparse AE) [Makhzani and Frey (2013)] performs top-k selection (i.e., pruning) on the hidden unit activations. The performance of the model is sensitive to the choice of hyperparameter $k$. To overcome this weakness, in [Makhzani and Frey (2013)], an empirically determined scheduling scheme for $k$ (during training) is introduced.

Top-k pruning operates on $x$ axis (Figure 3.5), promoting the population sparseness (Section 3.2.3). Sparse AE with $\ell_1$ norm penalty (on hidden unit activations) also enforces population sparseness, while placing importance on tuning the hyperparameter $\lambda$ to avoid the dead unit problem. But, unlike k-Sparse AE, this model has an external sparsity constraint which is reflected on its loss function (3.6) as an additional penalty term. Whereas, k-Sparse AE is an implicitly sparse model with its internal pruning mechanism (i.e., only preserving the top-k activations, and setting the rest of the hidden unit activations to zero) for enforcing sparsity. In other words, its loss function does not include any additional regularization or

penalty term, while the model pursues its reconstruction objective.

### 6.2.1 Architecture

For our experiments with k-Sparse AE, following the model configuration in [Makhzani and Frey (2013)], we use shallow, overcomplete, linear autoencoders with tied weights. Top-k selection introduces nonlinearity to the model, as it can also be seen as a variant of ReLU in which the threshold is the k-th largest activation (instead of zero).

As in previous chapters, all other components in the ASR pipeline (Figure 2.1), except for the k-Sparse AE, are implemented in Kaldi [Povey et al. (2011)]. k-Sparse AE is placed between the feature extractor and the acoustic model components, and is implemented using Pytorch [Paszke et al. (2017)] in full accordance with the training details in Section 5.2.2.

Following the proposed pipeline illustrated in Figure 5.3, we use 40 dimensional high-resolution MFCC features as input for the k-Sparse AE. To be consistent with the models in previous chapters, we set the number of hidden units in the encoding layer (denoted by $z$) as 1760 (i.e., $176 \times 10$, where 176 is the number of senones in the Kaldi AMI setup).

High-resolution MFCCs are normalized (via *apply-cmvn* command in Kaldi) before being fed to k-Sparse AE. Top-k selection on the encoding layer activations (i.e., encodings) is handled by Pytorch's *torch.topk* operation. When the activations from the hidden units outside the top-k support set (i.e., activations below the k-th largest activation) are set to zero, these units do not receive any feedback during backpropagation. However, thanks to Pytorch's *torch.topk* being a differentiable operation, backpropagation can be conducted without a problem (i.e., the entire computation graph is differentiable).

Unlike [Makhzani and Frey (2013)] that tunes $k$ empirically with a scheduler, we hypothesize that batch size $b_{size}$ (i.e., number of data samples in a mini-batch, or number of MFCC frames in our case) plays important role for setting hyperparameter $k$ and for setting the upper limit for the number of hidden units $n_{hu}$, such that $n_{hu} \leq k \times b_{size}$. This is to accommodate the extreme case where for each data sample in the batch, different sets of $k$ hidden units are active. In addition, $k \leq n_{hu}$ constitutes the upper limit for $k$.

$$k \leq n_{hu} \leq k \times b_{size} \tag{6.1}$$

So, for a given data sample, $k$, the number of units with non-zero activations among all hidden units in the encoding layer, can actually be reformulated as $k = n_{hu} \times c_{population}$, where $c_{\text{population}}$ denotes the population sparseness coefficient in [0,1] interval. For instance, given $n_{hu} = 1760$ and $c_{\text{population}} = 0.1$, then $k = 176$ (i.e., 176 units in 1760 dimensional encoding layer produce non-zero activation for any given input data sample). Based on this, reformulating (6.1),

$$n_{hu} \times c_{\text{population}} \leq n_{hu} \leq n_{hu} \times c_{\text{population}} \times b_{size} \tag{6.2}$$

$$\frac{1}{c_{\text{population}}} \leq b_{size} \tag{6.3}$$

Therefore, in our case, for batch size of 100, minimum population sparseness coefficient ($c_{\text{population}}$) is 0.01 and $k$ therefore should be minimum 18 ($k = 1760 \times c_{\text{population}}$). To set $k$, we conduct grid search on $c_{\text{population}} \in \{0.05, 0.1, 0.3, 0.5\}$, which is equivalent to $k \in \{88, 176, 528, 880\}$ for neural network training, respectively. The optimal $k$ values are determined based on the k-Sparse encodings' performance for the frame-level phone classification task on development sets. Accordingly, the $k$ is tuned as 176 ($c_{\text{population}} = 0.1$) and 528 ($c_{\text{population}} = 0.3$) for IHM and SDM, respectively.

### 6.2.2 Recognition Performance

Once k-Sparse AE is trained, the reconstructed MFCCs are fed to the LF-MMI acoustic model for training, and later, LF-MMI log-likelihoods from the acoustic model are sent to the Kaldi decoder for decoding. The word recognition performances are presented in Table 6.2. In addition, to assess the representative power of AE encodings, simple neural network based phone classifiers are trained, in full accordance with the implementation details presented in Section 5.3. The frame-level phone classification accuracies are reported in Table 6.1.

We hypothesize that hard-zeros in AE encodings (denoted as k-Sparse features in Table 6.1) enable the simple classifier to discriminate (i.e., track, distinguish) the patterns in the data with ease; hence, attaining better frame-level phone classification accuracies. In addition, the best performing k-Sparse AE model with respect to accuracies on the development sets are chosen to determine the optimal values for hyperparameter $k$. k-Sparse features in Table 6.1 and k-Sparse AE in Table 6.2 are based on the optimal model with $k = 176$ and $k = 528$ for IHM and SDM, respectively.

***Table 6.1:*** *The frame-level phone classification for accuracies (in %) on IHM and SDM evaluation sets. The encodings from k-Sparse AE obtain the best performance on both close-field IHM and far-field SDM data. To present the full picture, we also report the performance for the encodings from shallow overcomplete sparse AEs in Chapter 5.*

| Architecture | $IHM_{accuracy}$ | $SDM_{accuracy}$ |
|---|---|---|
| Baseline | 39.2 | 33.7 |
| Sparse AE ($\lambda > 0$) | 38.0 | 33.5 |
| k-Sparse AE | 40.7 | 34.3 |

Indeed, k-Sparse features help the simple phone classifier to classify the patterns with greater success. For instance, their phone classification accuracy is greater than the case of the original

MFCC features as input (denoted as Baseline) for training the simple phone classifiers.

***Table 6.2:*** *The recognition performance (in WER%) for baseline LF-MMI system and the proposed approach with k-Sparse AE on IHM and SDM evaluation sets. To present a comparative view, we also report the performance of shallow overcomplete sparse AEs from Chapter 5 (denoted as Sparse AE ($\lambda > 0$)).*

| Architecture | $IHM_{WER}$ | $SDM_{WER}$ |
|---|---|---|
| Baseline | 19.2 | 41.1 |
| Sparse AE ($\lambda > 0$) | 19.7 | 40.5 |
| k-Sparse AE | 19.9 | 44.3 |

The improvement on phone classification accuracy does not reflect on the word recognition performance (Table 6.2). Given our findings in [Kabil and Bourlard (2022a)], this is due to the LF-MMI acoustic model's sensitivity to the distortion in the value range of the reconstructed MFCCs introduced by the internal sparsity mechanism (i.e., top-k selection in k-Sparse AE).

### 6.2.3 Analysis on High-dimensional Sparse Features

In this section, we first examine if k-Sparse AE behaves as expected and produces hard-zeroed activations. Along this line, to evaluate the sparsity of the hidden unit activations (i.e., encodings), we utilize $\ell_0$ and $\ell_0^\epsilon$ (introduced in Section 3.2.5). We then investigate if the encoding space (i.e., representation space) exhibits orthogonalization and/or smooth latent space properties (Section 3.2.5).

Note that we here follow the *same* methodology for evaluating the sparsity of activations (i.e., with same sparseness measures on the same randomly chosen subset) in Section 5.4.1 and for analyzing the senone subspaces (i.e., same fingerprint extraction protocol, same subset of utterances) in Section 5.4.2.

#### Sparsity of Activations

Using $\ell_0$, $\ell_0^\epsilon$, and the *same* randomly chosen subset of utterances from the development sets (used also in Chapters 4 and 5), we measure the sparsity of the hidden unit activations (i.e., AE encodings) and examine if k-Sparse AEs are indeed capable of producing hard-zeroed activations.

The comparison of Tables 6.3 and 6.4 clearly shows that k-Sparse AEs indeed produce hard-zeroed and sparser encodings.

***Table 6.3:*** *The sparsity of the hidden unit activations based on the complement of $\ell_0$ measure for a subset of utterances from IHM and SDM development sets. For full comparison, we also present the sparsity of the hidden unit activations with $\ell_0^\epsilon$ measure for Sparse AE with $\lambda > 0$ on the same subsets from IHM and SDM in Table 6.4.*

| Architecture | IHM ($k = 176$) | SDM ($k = 528$) |
|---|---|---|
| $(\ell_0)^c$ (in %) | 90 | 70 |

***Table 6.4:*** *The sparsity of the hidden unit activations from Sparse AE with $\lambda > 0$ via the complement of $\ell_0$ measure for a subset of utterances from IHM and SDM development sets. For simplicity, instead of a total number of zero entries, we provide their percentage over all entries in the activation vectors.*

| | IHM ($\lambda = 0.0001$) | SDM ($\lambda = 0.0005$) |
|---|---|---|
| $(\ell_0^\epsilon)^c$ (in %) | 9 | 6 |

**Subspace Analysis**

With our proposed approach for sparse modeling (here, by means of k-Sparse AE), we aim for individual subspaces to lie in the common high dimensional representation space in a scattered manner (as shown in Figure 4.1). Our hypothesis is that this behavior can induce the separability of senones (or other speech units with different granularity, described in Section 2.1.4).

Based on our findings with colormaps in Chapters 4 and 5, we state that the hidden unit activation vectors for each particular senone has a distinctive pattern and shares a few common features with other senones that have similar phonetic and articulatory characteristics. Therefore, we here conduct phone-level analysis of hidden unit activations, which is anticipated to be more interpretable by humans.

Hence, focusing on the *same* subset of utterance from development sets (with 419 utterances and 112386 frames) and using Figure 5.9 and Figure 5.10, we examine the ratio of *robust* matches (over all matches) with respect to the changes in the threshold, the AE model and data on which fingerprint vectors are computed. The *robust match* indicates that matched fingerprint vectors map to similar phone labels in terms of manner and/or place of articulation. For the case of desired robust modeling, we expect the ratio of number of robust matches over all matches to be stable (in [0-1] scale), as the matching threshold decreases.

Sparse AE models with $\lambda > 0$ are previously reported in Tables 5.6 and 5.7 and presented once again in Tables 6.5 and 6.6 for completeness.

***Table 6.5:*** *Behavior of the matchings of the fingerprints computed with autoencoders trained on IHM with respect to the changes in matching threshold, AE model (i.e., Sparse AE with $\lambda > 0$ and k-Sparse AE), and data. The cross-database analysis in Chapter 5 (i.e., IHM model-SDM data or vice versa) implies that SDM data might not be as distorted (e.g., reverberant) as we initially anticipated. Therefore, we conduct an additional round of analysis on noisy SDM data (i.e., SDM data with additive 5dB babble noise from Noisex dataset [Varga and Steeneken (1993)]. Note that the robust matches are determined based on the tables presented in Figures 5.9 and 5.10.*

| Threshold | IHM ($\lambda = 0.0001$) | IHM ($k = 176$) | Data |
|---|---|---|---|
| 1.0 | 1.0 | 1.0 | IHM |
| | 1.0 | 1.0 | SDM |
| | 1.0 | 1.0 | SDM+babble noise(SNR=5dB) |
| 0.99 | 0.91 | 0.74 | IHM |
| | 0.36 | 0.43 | SDM |
| | 0.41 | 0.35 | SDM+babble noise(SNR=5dB) |
| 0.98 | 0.45 | 0.41 | IHM |
| | 0.44 | 0.35 | SDM |
| | 0.38 | 0.32 | SDM+babble noise(SNR=5dB) |

***Table 6.6:*** *The behavior of the matchings of the fingerprints computed with autoencoders trained on SDM with respect to the changes in matching threshold, AE model (i.e., Sparse AE with $\lambda > 0$ and k-Sparse AE), and data. The cross-database analysis in Chapter 5 (i.e., IHM data-SDM model or vice versa) implies that SDM data might not be as distorted (e.g., reverberant) as we initially anticipated. Therefore, we conduct an additional round of analysis on noisy SDM data (i.e., SDM data with additive 5dB babble noise from Noisex dataset [Varga and Steeneken (1993)]. Note that the robust matches are determined based on the tables presented in Figures 5.9 and 5.10.*

| Threshold | SDM ($\lambda = 0.0005$) | SDM ($k = 528$) | Data |
|---|---|---|---|
| 1.0 | 1.0 | 1.0 | IHM |
| | 1.0 | 1.0 | SDM |
| | 1.0 | 1.0 | SDM+babble noise(SNR=5dB) |
| 0.99 | 1.0 | 0.82 | IHM |
| | 0.86 | 0.56 | SDM |
| | 0.59 | 0.53 | SDM+babble noise(SNR=5dB) |
| 0.98 | 0.91 | 0.52 | IHM |
| | 0.57 | 0.42 | SDM |
| | 0.51 | 0.48 | SDM+babble noise(SNR=5dB) |

As depicted in Tables 6.5 and 6.6, we observe that:

1. The AE models trained on SDM data are more robust when the matching threshold is decreased.
2. Compared to k-Sparse AEs, Sparse AE models (with $\lambda > 0$) demonstrate robust characteristics with higher robust match ratio, especially in low(er) threshold cases.
3. Among all datasets and models, Sparse AEs trained on SDM data (i.e, SDM $\lambda = 0.0005$ in Table 6.6 obtains the most robust performance, with higher robust match ratio throughout decreasing trend of matching threshold.
4. The cross-database analysis (i.e., IHM data-SDM model or vice versa) implies that SDM data might not be as distorted (e.g., reverberant) as we initially anticipated. This is also in line with the findings presented in [Tang et al. (2018)] and [Kabil and Bourlard (2022a)].

## 6.3 Winner-Take-All Autoencoders (WTA)

Winner-Take-All (WTA) [Makhzani and Frey (2015)] is a computational principle utilized in neural networks so that the neurons (in the same network layer) compete with each other for activation [Srivastava et al. (2013)]. WTA networks are commonly used in the computational models of the brain, particularly for distributed decision making and action selection in the cortex [Carpenter and Grossberg (1987), Riesenhuber and Poggio (1999), Hawkins et al. (2017)]. Being biologically inspired models, Winner-Take-All autoencoders (WTA AEs) [Makhzani and Frey (2015)] use mini-batch statistics to maintain life-time sparseness on the hidden unit activations.

Just like k-Sparse AE (Section 6.2), WTA AE takes advantage of the computational principles in competitive learning. However, unlike k-Sparse AE, in high sparsity conditions, WTA AE does not suffer from the dead unit problem. Instead, it simply enforces life-time sparseness over the hidden units (Figure 3.5). That is, WTA AE focuses on the activation pattern of each individual hidden unit during its life-time (e.g., during neural network training) while still maintaining sparse representations. In other words, the hidden units (i.e., neurons in the encoding layer) are restricted from being active or inactive (e.g., on or off) all times. This simply encourages the hidden units to specialize, representing certain characteristics in the training data.

### 6.3.1 Architecture

For our experiments, following [Makhzani and Frey (2015)], we use shallow overcomplete WTA AE with untied weights and ReLU activation on the hidden units. In addition, we impose life-time sparseness by keeping the k largest activation of each individual hidden unit across the data samples in the mini-batch, and setting the rest of the activations to zero.

As in previous chapters, all other components in the ASR pipeline (Figure 2.1), except for the WTA AE, are implemented in Kaldi [Povey et al. (2011)]. WTA AE is placed between the feature extractor and the acoustic model components, and is implemented using Pytorch [Paszke et al. (2017)] in full accordance with the training details in Section 5.2.2.

Following the proposed pipeline illustrated in Figure 5.3, we use 40 dimensional high-resolution MFCC features as input to the WTA AEs. To be consistent with the models in previous chapters, we set the number of hidden units in the encoding layer as 1760 (i.e., $176 \times 10$, where 176 is the number of senones in the Kaldi AMI setup).

High-resolution MFCCs (produced by the feature extractor) are normalized (via *apply-cmvn* command in Kaldi) before being sent to WTA AEs. Top-k selection on the encoding layer activations (i.e., encodings) is handled by Pytorch's *torch.topk* operation. In the backpropagation phase, we only propagate the error through the units with non-zero activations, thanks to Pytorch's differentiable topk operation *torch.topk(), with dim=1*.

Hyper-parameter $k$ plays a crucial role for WTA AE performance. As the pruning (i.e., top k operation on activations) actually relies on the mini-batch statistics, we hypothesize that batch size $b_{size}$ can point the implicit relations between the number of hidden units $n_{hu}$ in the encoding layer and hyper-parameter $k$, such that $b_{size} \leq k \times n_{hu}$. This is due to the realization of the extreme case where each hidden unit is active for different data samples in the mini-batch (i.e., active in absolutely non-overlapping manner). In addition, for obvious reasons, $k \leq b_{size}$.

$$k \leq b_{size} \leq n_{hu} \times k \tag{6.4}$$

So, given the data samples in the mini-batch, $k$ denotes the number of times a hidden unit is active. It can actually be reformulated as $k = b_{size} \times c_{\text{lifetime}}$, where $c_{\text{lifetime}}$ denotes life-time sparseness coefficient in [0,1] interval. For instance, given $b_{size} = 100$ and $c_{\text{lifetime}} = 0.1$, $k = 10$ (i.e., each individual hidden unit is active for 10 out of 100 times, or 10 data sample out of mini-batch with 100 data samples). Based on this, (6.4) can be restated as,

$$b_{size} \times c_{\text{lifetime}} \leq b_{size} \leq b_{size} \times c_{\text{lifetime}} \times n_{hu} \tag{6.5}$$

$$\frac{1}{c_{\text{lifetime}}} \leq n_{hu} \tag{6.6}$$

Therefore, in our case, given $n_{hu} = 1760$, the minimum lifetime sparseness coefficient ($c_{\text{lifetime}}$) is 0.0006, regardless of the batch size. To set $k$, we conduct search on $c_{\text{lifetime}} \in \{0.001, 0.01, 0.05, 0.1, 0.3, 0.5\}$, which is equivalent to $k$ as $\{1, 5, 10, 30, 50\}$ for neural network training, respectively. The optimal $k$ values are determined based on the WTA encodings' performance for the frame-

level phone classification task on development sets. Accordingly, the $k$ is tuned as 10, for both IHM and SDM.

### 6.3.2 Recognition Performance

Once WTA AE training is finished, the reconstructed MFCCs are sent to the LF-MMI acoustic model for training, and later senone log-likelihoods from the acoustic model are sent to the Kaldi decoder for decoding. The word recognition performances are presented in Table 6.8. In addition, to assess the representative power of AE encodings, simple neural network based phone classifiers are trained, in full accordance with the implementation details presented in Section 5.3. The frame-level phone classification accuracies are reported in Table 6.7.

We hypothesize that hard-zeros in AE encodings (denoted as WTA features in Table 6.7) is to enable the simple classifier to discriminate (i.e., track, distinguish) the patterns in the data in ease; hence, reaching to better frame-level phone classification accuracies. In addition, the best performing WTA AE model with respect to the accuracies on the development sets are chosen to determine the optimal values for hyper-parameter $k$. WTA features in Table 6.1 and WTA AE in Table 6.2 are based on the optimal model with $k = 10$ for both IHM and SDM.

***Table 6.7:*** *The frame-level phone classification accuracy (in %) on IHM and SDM evaluation sets. The encodings from WTA AE obtain the best performance on far-field SDM data. For the sake of completeness, we also report the performance for the encodings from shallow overcomplete sparse AEs from Chapter 5.*

| Architecture | $IHM_{accuracy}$ | $SDM_{accuracy}$ |
|---|---|---|
| Baseline | 39.2 | 33.7 |
| Sparse AE ($\lambda > 0$) | 38.0 | 33.5 |
| WTA AE | 38.4 | 35.2 |

Indeed, encodings from WTA AE help the simple phone classifier to classify the patterns with greater success for far-field SDM case. The phone classification accuracy is greater than the case of the original MFCC features (denoted as Baseline) as input of the simple phone classifier.

***Table 6.8:*** *The recognition performance (in WER%) for baseline LF-MMI system and the proposed approach with WTA AE on IHM and SDM evaluation sets. To present a comparative view, we also report the performance of shallow overcomplete sparse AEs from Chapter 5 (denoted as Sparse AE ($\lambda > 0$)).*

| Architecture | $IHM_{WER}$ | $SDM_{WER}$ |
|---|---|---|
| Baseline | 19.2 | 41.1 |
| Sparse AE ($\lambda > 0$) | 19.7 | 40.5 |
| WTA AE | 19.8 | 42.1 |

The improvement on phone classification accuracy does not reflect on the word recognition performance (Table 6.8). Given our findings in [Kabil and Bourlard (2022a)], this is due to the LF-MMI acoustic model's sensitivity to the distortion in the value range of the reconstructed MFCCs introduced by the internal sparsity mechanism. However, in terms of word recognition, WTA AE outperforms k-Sparse AE, which boosts the population sparseness (Section 3.2.4), as expected.

### 6.3.3 Analysis on High-dimensional Sparse Features

In this section, we first examine if WTA AE behaves as expected and produces hard-zeroed activations. Along this line, to evaluate the sparsity of the hidden unit activations (i.e., encodings), we utilize $\ell_0$ and $\ell_0^\epsilon$ (introduced in Section 3.2.5). We then investigate if the encoding space (i.e., representation space) exhibits orthogonalization and/or smooth latent space properties (Section 3.2.5).

Note that we here follow the *same* methodology for evaluating the sparsity of activations (i.e., with same sparseness measures on the same randomly chosen subset) in Section 5.4.1 and for analyzing the senone subspaces (i.e., same fingerprint extraction protocol, same subset of utterances) in Section 5.4.2.

#### Sparsity of Activations

Using $\ell_0$ and $\ell_0^\epsilon$, and focusing on the *same* randomly chosen subset of utterances from the development sets (used also in Chapters 4 and 5), we measure the sparsity of the hidden unit activations (i.e., AE encodings) and examine if WTA AEs are indeed capable of producing hard-zeroed activations.

The comparison of Tables 6.4 and 6.9 clearly shows that WTA AEs indeed produce hard-zeroed and sparser encodings.

***Table 6.9:*** *The sparsity of the hidden unit activations based on the complement of $\ell_0$ measure on subset of utterances from IHM and SDM development sets. For simplicity, instead of a total number of zero entries, we provide their percentage over all entries in the activation vectors for (randomly chosen) subset of utterances. For full comparison, we also present the sparsity of the hidden unit activations with $\ell_0^\epsilon$ measure for Sparse AE with $\lambda > 0$ on the same subsets from IHM and SDM in Table 6.4.*

| Architecture | IHM ($k = 10$) | SDM ($k = 10$) |
|---|---|---|
| $(\ell_0)^c$ (in %) | 90 | 90 |

### Subspace Analysis

With our proposed approach for sparse modeling (here, by means of WTA AE), we aim for individual subspaces to lie in the common high dimensional representation space in a scattered manner (as shown in Figure 4.1). Our hypothesis is that this behavior can induce the separability of senones (or other speech units with different granularity, described in Section 2.1.4).

Based on our findings with colormaps in Chapters 4 and 5, we state that the hidden unit activation vectors for each particular senone has a distinctive pattern and shares a few common features with other senones that have similar phonetic and articulatory characteristics. Therefore, we here conduct phone-level analysis of hidden unit activations, which is anticipated to be more interpretable for humans.

Hence, focusing on the *same* randomly chosen subset of utterance from development sets with 419 utterances and 112386 frames) and using Figures 5.9 and 5.10, we examine the ratio of *robust* matches with respect to the changes in the threshold, the AE model and data on which fingerprint vectors are computed. With the decrease in threshold (i.e., matching condition is weakened), the number of matches and number of false matches increase, as expected. However, our expectation with robust modeling is to avoid non-robust false matches as much as possible, especially under low(er) matching thresholds.

Sparse AE models with $\lambda > 0$ are previously reported in Tables 5.6 and 5.7 and presented once again in Tables 6.10 and 6.11 for a complete overview.

***Table 6.10:*** *The behavior of the matchings (of the fingerprints computed with autoencoders trained on IHM) with respect to the changes in matching threshold, AE model (i.e., Sparse AE with $\lambda > 0$ and WTA AE), and data. The cross-database analysis (i.e., IHM model-SDM data or vice versa) in Chapter 5 implies that SDM data might not be as distorted (e.g., reverberant) as we initially anticipated. Therefore, we conduct an additional round of analysis on noisy SDM data (i.e., SDM data with additive 5dB babble noise from Noisex dataset [Varga and Steeneken (1993)]. Note that the robust matches are determined based on the tables presented in Figures 5.9 and 5.10.*

| Threshold | IHM ($\lambda = 0.0001$) | IHM ($k = 10$) | Data |
|---|---|---|---|
| 1.0 | 1.0 | 1.0 | IHM |
| | 1.0 | 1.0 | SDM |
| | 1.0 | 1.0 | SDM + babble noise (SNR=5dB) |
| 0.99 | 0.91 | 1.0 | IHM |
| | 0.36 | 1.0 | SDM |
| | 0.41 | 0.95 | SDM+babble noise(SNR=5dB) |
| 0.98 | 0.45 | 0.97 | IHM |
| | 0.44 | 0.96 | SDM |
| | 0.38 | 0.93 | SDM+babble noise(SNR=5dB) |

***Table 6.11:*** *The behavior of the matchings (of the fingerprints computed with autoencoders trained on SDM) with respect to the changes in matching threshold, AE model (i.e., Sparse AE with $\lambda > 0$ and WTA AE), and data. The cross-database analysis (i.e., IHM data-SDM model or vice versa) in Chapter 5 implies that SDM data might not be as distorted (e.g., reverberant) as we initially anticipated. Therefore, we conduct an additional round of analysis on noisy SDM data (i.e., SDM data with additive 5dB babble noise from Noisex dataset [Varga and Steeneken (1993)]. Note that the robust matches are determined based on the tables presented in Figures 5.9 and 5.10.*

| Threshold | SDM ($\lambda = 0.0005$) | SDM ($k = 10$) | Data |
|---|---|---|---|
| 1.0 | 1.0 | 1.0 | IHM |
| | 1.0 | 1.0 | SDM |
| | 1.0 | 1.0 | SDM+babble noise(SNR=5dB) |
| 0.99 | 1.0 | 1.0 | IHM |
| | 0.86 | 1.0 | SDM |
| | 0.59 | 1.0 | SDM+babble noise(SNR=5dB) |
| 0.98 | 0.91 | 0.98 | IHM |
| | 0.57 | 0.95 | SDM |
| | 0.51 | 0.94 | SDM+babble noise(SNR=5dB) |

From Tables 6.5, 6.10 and Tables 6.6, 6.11, we observe that:

1. The AE models trained on SDM data are more robust when the matching threshold is decreased.

2. Compared to k-Sparse AEs and Sparse AE models (with $\lambda > 0$), WTA AE models demonstrate the better modeling characteristics with higher robust match ratios, especially in low(er) threshold cases in the presence of additive noise.

3. Among all datasets and AE models, WTA AE trained on SDM data (i.e, SDM $k = 10$ in Table 6.11) obtains the most robust performance.

4. The cross-database analysis (i.e., IHM data-SDM model or vice versa) implies that SDM data might not be as distorted (e.g., reverberant) as we initially anticipated. This is also in line with the findings presented in [Tang et al. (2018)] and [Kabil and Bourlard (2022a)].

## 6.4 Conclusion

In the previous chapter (Chapter 5), with our experiment and analysis on high-dimensional AE encodings, we obtained improvements on WER, which encouraged us to use MFCCs as feature set in the rest of the thesis. However, we also observed that with MFCCs, it was more challenging to detect patterns in senone (and even phone) subspaces (compared to Chapter 4), as MFCCs are less task-driven, more entangled and harder for shallow overcomplete sparse autoencoders with $\ell_1$ norm penalty to process and produce close-to-zero encodings.

Therefore, in this chapter, we further explored sparse autoencoders with MFCCs as input features for speech recognition and modeling. We focused on sparse autoencoders with internal sparsity mechanisms, essentially k-Sparse autoencoders and Winner-Take-All autoencoders. Owing to their biologically inspired internal sparsity mechanisms, we found that these models promote hard-zeroed embeddings in the encoding layer, especially for the sake of interpretability of the subspace analysis of fingerprint vectors computed on high-dimensional encodings.

With our experiments and analysis on high-dimensional AE encodings,

1. We obtained improvement on phone classification accuracy when hard-zeroed encodings from k-Sparse AE and WTA AE are used for frame-level phone classification for distorted far-field speech. This indicates that these encodings help the simple phone classifier to classify the patterns in the data.

2. We could not obtain improvement on WERs, due to the distortion in the value range of the reconstructed MFCCs, probably because of the internal sparsity mechanism operating on hidden layer activations.

3. We observed that among all dataset and AE models, WTA AE trained on SDM data obtains the most robust performance, even in the presence of additive noise, especially when the matching threshold is decreased.

4. The cross-database analysis (i.e., IHM data-SDM model or vice versa) underlined that SDM data might not be as distorted (e.g., reverberant) as initially anticipated. This is also stated in [Tang et al. (2018)] and [Kabil and Bourlard (2022a)].

Therefore, in the next chapter, we investigate the generalization power of WTA AE models (trained on SDM data) while exploiting the model in the transfer learning scenarios for pathological speech recognition.

# 7 Transfer Learning for Pathological Speech Recognition

In this chapter [1], we repurpose the best performing implicitly constrained sparse autoencoder model from Chapter 6 in the transfer learning framework for pathological speech recognition. Winner-Take-All (WTA) autoencoder trained on far-field SDM data is the best performing model in Chapter 6 based on its robust speech modeling capacity and recognition performance. We will refer to this model as *WTA model* in the rest of this chapter.

In summary, we insert the WTA model between the feature extractor and the acoustic model components in the ASR pipeline which is built for pathological speech recognition task. For all transfer learning experiments, WTA model is kept fixed (i.e., frozen). We expect the fixed off-the-shelf WTA model to project the pathological acoustic features closer to the healthy control acoustic feature space. The projected acoustic features obtained from the WTA model are then sent to the acoustic model.

Based on acoustic model state, transfer learning experiments are conducted in *three scenarios*.

- In as-is scenario, projected acoustic features are sent the fixed LF-MMI acoustic model which is initially trained on healthy control speech.
- In finetuning scenario, the projected acoustic features are used for finetuning the LF-MMI acoustic model which is initially trained on healthy control speech.
- In training from scratch scenario, the projected acoustic features are used for training a brand new LF-MMI acoustic model with the same model configuration for the baseline acoustic model for healthy control speech.

Despite LF-MMI being sensitive to the changes in the value range of the input data, we obtained promising improvement in pathological speech recognition. Overall, our findings

[1] This chapter is partially based on :
Kabil, S.H., and Bourlard, H. (2022). From undercomplete to sparse overcomplete autoencoders to improve lf-mmi speech recognition. In Proceedings of Interspeech Conference.

point to WTA autoencoder's potential for generating features which can also be beneficial for other downstream tasks.

## 7.1 Introduction

One of the goals in our thesis was to learn meaningful and invariant speech representations (e.g., sparse distributed representations Section 3.2.3) by means of sparse autoencoders for different tasks ranging from speech modeling to recognition. Along this line, we repurpose the best performing implicitly constrained sparse autoencoder model from Chapter 6 in transfer learning framework for pathological speech recognition. In other words, in this chapter, we examine if these implicitly constrained sparse autoencoder models are capable of producing meaningful representations which can also be beneficial for downstream tasks like pathological speech recognition.

In Chapter 6, based on robust speech modeling capacity and recognition performance, Winner-Take-All (WTA) autoencoder trained on far-field SDM data is the best performing model. We will refer to this model as **WTA model** in the rest of the present chapter. In addition, our findings presented in Chapter 6 demonstrate that the SDM dataset is not as reverberant as we initially anticipated. In fact, its distortion is probably not due to the reverberation, but to some other factors, such as cross-talking [Tang et al. (2018)]. Furthermore, we observe that the models trained on SDM data have better generalization power, implying that SDM contains more acoustic variation compared to close-field IHM data. This is also in line with our findings in [Kabil and Bourlard (2022a)].

Proposed transfer learning framework with pre-trained WTA model can really be useful for improving the recognition performance. Actually, the use of autoencoders in transfer learning setup for speech processing tasks, such as speech recognition [Feng et al. (2014), Grozdić and Jovičić (2017)], feature enhancement [Ishii et al. (2013), Tang et al. (2018)] and speaker recognition [Plchot et al. (2016), Shon et al. (2017)] have been extensively explored in literature before. And, accordingly we hypothesize that the WTA model can reconstruct pathological data in such a way that its subspace components become more convenient for the LF-MMI acoustic model, which is trained on healthy speech, to process or to be trained.

## 7.2 Our Approach

Pathological speech constitutes several challenges. It is largely distorted (i.e., acoustically mismatched) and hard to access, mainly due to legal and ethical requirements. Therefore, there does not exist numerous pathological speech datasets. As anticipated, the acoustic models trained on pathological speech lack generalization power, due to the small amount of available pathological data. Whereas, the acoustic models trained on easy-to-access healthy control speech do not provide improvement on WER, when used for pathological speech recognition, mainly due to the mismatches between healthy and pathological speech acoustic

spaces.

To overcome this challenge, we here present our transfer learning framework. We insert an off-the-shelf WTA model between the feature extractor and the acoustic model components in the ASR pipeline (Figure 2.1). We expect the fixed, off-the-shelf WTA model to project the pathological acoustic space closer to healthy control acoustic space. In other words, the healthy acoustic space knowledge learned by the WTA model in Chapter 6 is transferred. This is expected to improve the pathological speech recognition performance, especially when the acoustic models trained on healthy control speech are used for decoding pathological speech.

With this pipeline, based on the state of the acoustic model, transfer learning experiments are conducted in three scenarios. Note that during all experiments, the WTA model is kept fixed.

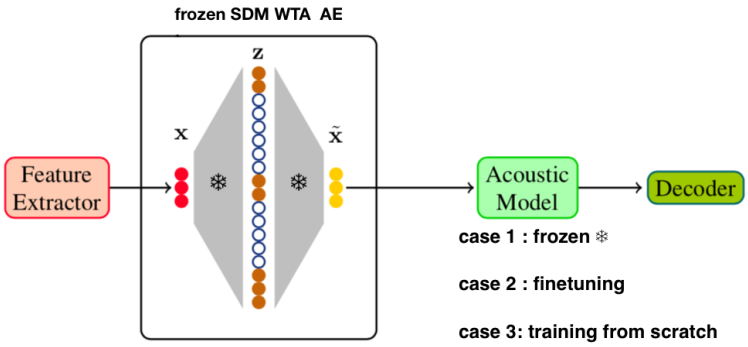



***Figure 7.1:*** *Transfer learning framework for pathological speech recognition.*

In **as-is scenario**, we pass the pathological MFCCs to the fixed WTA model and obtain the reconstructed features. These reconstructed MFCCs are then sent to the fixed LF-MMI acoustic model initially trained on healthy control speech.

In **finetuning scenario**, we feed frozen WTA model with the pathological MFCCs to get reconstructions for for finetuning the LF-MMI acoustic model which is previously trained on healthy control speech.

In **training from scratch scenario**, we use the reconstructed pathological MFCCs from the WTA model for training a brand-new LF-MMI acoustic model.

## 7.3 Experimental Results

We insert an off-the-shelf WTA model between the feature extractor and the acoustic model components in the ASR pipeline (Figure 2.1). We expect this off-the-shelf WTA model to project the pathological speech closer to healthy speech so that easy-to-access acoustic models trained on healthy speech can actually be useful for recognition and reach better WER for pathological speech recognition.

In all transfer learning scenarios, the same model configuration (Section 2.5) is used for LF-MMI acoustic models. We use DYS and CTL subsets from the UA-Speech dataset (Section 2.3.2) for pathological speech and healthy control speech, respectively.

Following the model configuration used in [Hermann and Doss (2020)] and described in Section 2.5, the baseline WERs (i.e., without the involvement of WTA model in ASR pipeline) is reported in Table 7.1.

***Table 7.1:*** *The recognition performance (in WER%) for baseline LF-MMI acoustic models trained on CTL and DYS portions of UA-Speech database. Mainly, because of the acoustic mismatch between the healthy control speech (denoted as CTL) and pathological speech (denoted as DYS), the LF-MMI acoustic model trained on CTL performs poorly for DYS recognition.*

| Train on | Decode on | WER |
|---|---|---|
| CTL | CTL | 18.5 |
| DYS | DYS | 38.9 |
| CTL | DYS | **62.5** |

With our proposed framework, we expect to improve the baseline WER (**62.5%**) reported in Table 7.1. In addition, to evaluate the impact of involvement of the LF-MMI acoustic model in the proposed transfer learning framework, we devise three cases.

### 7.3.1 As-is

In **as-is case** (case 1 in Figure 7.1), we feed the pathological MFCCs to the WTA model and obtain the reconstructed features. These reconstructed MFCCs are then fed to the **frozen** LF-MMI acoustic model which is initially trained on healthy control speech. The fact that we manage to improve the baseline WER slightly (i.e., close to 2% absolute improvement) implies that our hypothesis for transferring the healthy acoustic knowledge is promising.

***Table 7.2:*** *The recognition performance (in WER%) for LF-MMI systems trained on CTL in as-is scenario.*

| Model | WER |
|---|---|
| Baseline | 62.5 |
| Case 1: with frozen CTL acoustic model | 61.0 |

### 7.3.2 Finetuning

In **finetuning case** (case 2 in Figure 7.1), we feed the pathological MFCCs to WTA model and use the reconstructed features for **finetuning** (i.e., further train) the LF-MMI acoustic model which is previously trained on healthy control speech. We observe slight degradation in the performance, which is in line with our findings in [Kabil and Bourlard (2022a)]. Exploring the

hyperparameter space (e.g., number of epochs, batch size) plays crucial role in finetuning when adapting LF-MMI acoustic model on projected features.

***Table 7.3:*** *The recognition performance (in WER%) for LF-MMI systems trained on CTL in finetuning scenario.*

| Model | WER |
|---|---|
| Baseline | 62.5 |
| Case 2: finetuning CTL acoustic model | 63.0 |

### 7.3.3 Training from Scratch

In **training from scratch case** (case 3 in Figure 7.1), we get the reconstructed pathological MFCCs from the WTA model. We then use these features for **training** a brand-new LF-MMI acoustic model from scratch with the same model configuration as the baseline acoustic model for healthy control speech. Along the same line with our findings in [Kabil and Bourlard (2022a)], we observe that exploring hyperparameter space has an impact on the recognition performance. Despite our inability for proper exploration of the search space because of time limitations, we still manage to improve the WER (i.e., close to 3%).

***Table 7.4:*** *The recognition performance (in WER%) for LF-MMI systems trained on CTL in training from scratch scenario.*

| Model | WER |
|---|---|
| Baseline | 62.5 |
| Case 3: training new acoustic model | 59.7 |

## 7.4 Conclusion

In this chapter, we repurposed the best performing implicitly constrained sparse autoencoder model from Chapter 6 in the transfer learning framework for pathological speech recognition. Basically, we inserted off-the-shelf WTA model between the feature extractor and the acoustic model components in the ASR pipeline (Figure 2.1). Our goal was to project pathological (DYS) acoustic space close to the healthy control (CTL) acoustic space so that easier-to-access acoustic models trained on healthy control speech can actually be useful for pathological speech recognition.

Keeping the off-the-shelf WTA model from Chapter 6 fixed, based on the state of the acoustic model, we devised three scenarios for transfer learning experiments. In as-is scenario, projected pathological acoustic features were sent the fixed LF-MMI acoustic model which was initially trained on healthy control speech. In finetuning scenario, the projected pathological acoustic features were used for finetuning the LF-MMI acoustic model which was initially

trained on healthy control speech. In training from scratch scenario, the projected pathological acoustic features (i.e., reconstructions from fixed WTA model) were used for training a brand new LF-MMI acoustic model with the same model configuration for the baseline acoustic model for healthy control speech.

With the presented transfer learning framework, easy-to-access CTL acoustic models became applicable for decoding pathological DYS speech. We managed to obtain performance improvements on WER, despite the sensitivity of the finely-tuned LF-MMI acoustic models. This implies that WTA autoencoder models are promising for learning generalizable features.

# 8 Conclusion and Directions for Future Work

In this chapter, Section 8.1 summarizes the conclusions of this thesis and Section 8.2 discusses the directions for future research.

## 8.1 Conclusions

In this thesis, we addressed the problem of learning informative, interpretable and generalizable speech representations using deterministic sparse autoencoders. Based on speech production knowledge, we hypothesize that informative speech components live in class-specific low-dimensional subspaces whereas random unstructured noise is scattered in high dimensions. In this regard, we identified shallow overcomplete sparse autoencoders as excellent networks for modeling the class-specific low-dimensional subspaces of speech, holding the key for robust ASR and thorough understanding for speech modeling.

Taking [Dighe (2019)] as our reference point, we proposed the use of shallow overcomplete sparse autoencoders for sparse modeling of speech data. For speech recognition, we inspected the reconstructed features obtained from the output layer of the autoencoder. Sparsity constraint was enforced on the autoencoder model. Hence, we expected the reconstructions from the output of the model to be better features for acoustic model training, compared to their original counterparts. For speech modeling, we examined the high-dimensional sparse features obtained from the encoding layer of the autoencoder. We expected that high-dimensional sparse encoding space to satisfy the smooth latent space properties. In addition, with different sparsity constraints, we aimed to learn sparse distributed representations that are task-agnostic and therefore useful for different speech tasks.

First, we exploited duality between the dictionary learning [Candès and Wakin (2008)] and shallow overcomplete sparse autoencoders with $\ell_1$ norm penalty on hidden unit activations. In other words, we effectively transformed the dictionary learning problem to representation learning problem. For speech recognition experiments, we observed that acoustic features (i.e., MFCCs) constitute a more convenient feature set for improvements in terms of WER,

compared to acoustic model output features (i.e., LF-MMI log-likelihoods).

For speech modeling, we modeled the representation space where each sparse encoding resides in a similar manner to the feature vectors in vector space models used in information retrieval and natural language processing. In the vector space model, cosine similarity is generally used as metric. We showed that encodings which were matched together based on cosine similarity, shared similar articulatory configurations.

We then explored implicitly constrained sparse autoencoders for better speech modeling and learning sparse distributed representations, which are expected to be task-agnostic and therefore, could be useful for different speech tasks. With k-Sparse autoencoders [Makhzani and Frey (2013)] and Winner-Take-All (WTA) autoencoders [Makhzani and Frey (2015)], we studied the impact of adopting different viewpoints to sparsity in the model configuration. We observed that WTA autoencoders were the best performing models in terms of robust modeling of speech units, even under the presence of additive noise.

Finally, we repurposed the pre-trained WTA autoencoder to benefit its generalization power in the transfer learning framework for pathological speech recognition task. Mainly, we desired to exploit the acoustic knowledge, which WTA autoencoder learned on far-field speech, to project the pathological acoustic space closer to the healthy control acoustic space. With this, easy-to-access healthy control acoustic models became applicable on pathological speech data, and we could obtain promising improvements on the pathological speech recognition performance.

## 8.2 Directions for Future Research

The research presented in this thesis can be further extended along the following lines, including:

- As explained in Chapters 4 and 5, Kaldi introduced some challenges while building the ASR pipeline with additional AE component. Using another toolkit with flexibility (e.g., SpeechBrain toolkit [Ravanelli et al. (2021)]), the exploration of joint training of the acoustic model and AEs or multi-task learning across different speech tasks are doable.
- Inspired by the self-supervised learning [Liu et al. (2022)], the proposed approach in Chapter 4 can also be performed using different loss functions (e.g., triplet loss [Jansen et al. (2018)]) in the sparse autoencoder configuration.
- In this thesis, we only focused on deterministic sparse autoencoders. However, the same research can also be conducted on different sparse autoencoder configurations. For instance, deep WTA AE would be beneficial for understanding the impact of sparsity in different layers of the network. To understand the impact of contextual data, sequence-to-sequence autoencoders [Amiriparian et al. (2017)] or CNN-based autoencoders [Masci et al. (2011)] with sparsity constraints can be explored. In addition, as

alternative to deterministic models, variational sparse autoencoders [Asperti (2018)] can also be adopted.

- The downstream tasks for transfer learning study in Chapter 7 can also be diversified. For instance, in the case of accented speech, transformation of the accented acoustic space to native acoustic space may lead to improvements for accented speech recognition task. In addition, at this point, it is important to keep in mind that dataset choice can play a decisive role for the performance.

# Bibliography


Ahmad, S. and Scheinkman, L. (2019). How can we be so dense? the benefits of using highly sparse representations. Technical report, Numenta Inc.

Allen, J. B. (1995). How do humans process and recognize speech? In *Modern methods of speech processing*. Springer.

Amiriparian, S., Freitag, M., Cummins, N., and Schuller, B. (2017). Sequence to sequence autoencoders for unsupervised representation learning from audio. In *DCASE*.

Asperti, A. (2018). Sparsity in variational autoencoders. *arXiv preprint arXiv:1812.07238*.

Bengio, Y. (2009). *Learning deep architectures for AI*. Now Publishers Inc.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*.

Bengio, Y., Lecun, Y., and Hinton, G. (2021). Deep learning for ai. *Communications of the ACM*.

Bourlard, H. and Kabil, S. H. (2022). Autoencoders reloaded. *Biological cybernetics*.

Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*.

Bourlard, H. and Morgan, N. (2012). *Connectionist speech recognition: a hybrid approach*, volume 247. Springer Science & Business Media.

Candès, E. and Wakin, M. (2008). An introduction to compressive sampling. *IEEE signal processing magazine*.

Carpenter, G. A. and Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer vision, graphics, and image processing*.

Chen, S., Donoho, D., and Saunders, M. (2001). Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*.

Chen, Y. and Zaki, M. J. (2017). Kate: K-competitive autoencoder for text. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Deng, L., Ramsay, G., and Sun, D. (1997). Production models as a structural basis for automatic speech recognition. *Speech Communication*.

Dighe, P. (2019). *Sparse and Low-rank Modeling for Automatic Speech Recognition*. PhD thesis.

Dighe, P., Asaei, A., and Bourlard, H. (2019). Low-rank and sparse subspace modeling of speech for dnn based acoustic modeling. *Speech Communication*.

Dosovitskiy, A. and Brox, T. (2016). Generating images with perceptual similarity metrics based on deep networks. *Advances in neural information processing systems*.

Feng, X., Zhang, Y., and Glass, J. (2014). Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition. In *IEEE international conference on acoustics, speech and signal processing (ICASSP)*.

Francis, W. N. and Kucera, H. (1979). Brown corpus manual. *Letters to the Editor*.

Frankel, J. and King, S. (2001). Asr-articulatory speech recognition.

Gemmeke, J. F., Virtanen, T., and Hurmalainen, A. (2011). Exemplar-based sparse representations for noise robust automatic speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*.

Georgescu, A.-L., Cucu, H., and Burileanu, C. (2019). Kaldi-based dnn architectures for speech recognition in romanian. In *2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*. IEEE.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings.

Golik, P., Tüske, Z., Schlüter, R., and Ney, H. (2015). Convolutional neural networks for acoustic modeling of raw time signal in lvcsr. In *Sixteenth annual conference of the international speech communication association*.

Golub, G. and Reinsch, C. (1971). *Linear Algebra, Singular value decomposition and least squares solutions*. Springer.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Grozdić, Đ. T. and Jovičić, S. T. (2017). Whispered speech recognition using deep denoising autoencoder and inverse filtering. *IEEE Transactions on Audio, Speech, and Language Processing*.

Gupta, K. and Majumdar, A. (2016). Sparsely connected autoencoder. In *2016 International Joint Conference on Neural Networks (IJCNN)*.

Hawkins, J., Ahmad, S., and Cui, Y. (2017). A theory of how columns in the neocortex enable learning the structure of the world. *Frontiers in neural circuits*.

Hermann, E. and Doss, M. M. (2020). Dysarthric speech recognition with lattice-free mmi. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*.

Hoyer, P. O. (2004). Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*.

Hurley, N. and Rickard, S. (2009). Comparing measures of sparsity. *IEEE Transactions on Information Theory*.

Hwang, M.-Y., Huang, X., and Alleva, F. A. (1996). Predicting unseen triphones with senones. *IEEE Transactions on speech and audio processing*.

Ishii, T., Komiyama, H., Shinozaki, T., Horiuchi, Y., and Kuroiwa, S. (2013). Reverberant speech recognition based on denoising autoencoder. In *Interspeech*.

Jansen, A. and Niyogi, P. (2006). Intrinsic fourier analysis on the manifold of speech sounds. In *IEEE International Conference on Acoustics Speech and Signal Processing Proceedings (ICASSP)*.

Jansen, A., Plakal, M., Pandya, R., Ellis, D. P., Hershey, S., Liu, J., Moore, R. C., and Saurous, R. A. (2018). Unsupervised learning of semantic audio representations. In *EEE international conference on acoustics, speech and signal processing (ICASSP)*.

Jelinek, F. (1998). *Statistical methods for speech recognition*. MIT press.

Jolliffe, I. (1986). *Principal Component Analysis, Springer Series in Statistics (2nd edition)*. Springer-Verlag New York.

Kabil, S. H. and Bourlard, H. (2022a). From undercomplete to sparse overcomplete autoencoders to improve lf-mmi speech recognition. In *Interspeech*.

Kabil, S. H. and Bourlard, H. (2022b). Sparse autoencoders to enhance speech recognition. Idiap technical report.

Kabil, S. H. and Bourlard, H. (2022c). Speech modeling using sparse autoencoders. Idiap technical report.

Kanerva, P. (1988). *Sparse distributed memory*. MIT press.

King, S., Frankel, J., Livescu, K., McDermott, E., Richmond, K., and Wester, M. (2007). Speech production knowledge in automatic speech recognition. *The Journal of the Acoustical Society of America*.

Kingma, D. P. and Welling, M. (2019). An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*.

Liu, S., Mallol-Ragolta, A., Parada-Cabeleiro, E., Qian, K., Jing, X., Kathan, A., Hu, B., and Schuller, B. W. (2022). Audio self-supervised learning: A survey. *arXiv preprint arXiv:2203.01205*.

Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2009). Online dictionary learning for sparse coding. In *International Conference on Machine Learning (ICML)*.

Makhzani, A. and Frey, B. (2013). K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*.

Makhzani, A. and Frey, B. J. (2015). Winner-take-all autoencoders. *Advances in neural information processing systems*.

Masci, J., Meier, U., Cireşan, D., and Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. In *International conference on artificial neural networks*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*.

Ng, A. et al. (2011). Sparse autoencoder. *CS294A Lecture notes*.

Ngiam, J., Chen, Z., Bhaskar, S. A., Koh, P. W., and Ng, A. Y. (2011). Sparse filtering. In *Advances in Neural Information Processing Systems*.

Olshausen, B. A. and Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*.

Olshausen, B. A. and Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*.

Paszke, A., Gross, S., Chintala, S., and Chanan, G. (2017). Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration.

Peddinti, V., Povey, D., and Khudanpur, S. (2015). A time delay neural network architecture for efficient modeling of long temporal contexts. In *Sixteenth annual conference of the international speech communication association*.

Plchot, O., Burget, L., Aronowitz, H., and Matejka, P. (2016). Audio enhancing with dnn autoencoder for speaker recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., et al. (2011). The kaldi speech recognition toolkit. In *Proceedings of the IEEE 2011 workshop on automatic speech recognition and understanding*.

Povey, D., Kanevsky, D., Kingsbury, B., Ramabhadran, B., Saon, G., and Visweswariah, K. (2008). Boosted mmi for model and feature-space discriminative training. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Povey, D., Peddinti, V., Galvez, D., Ghahremani, P., Manohar, V., Na, X., Wang, Y., and Khudanpur, S. (2016). Purely sequence-trained neural networks for asr based on lattice-free mmi. In *Interspeech*.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*.

Rath, G., Guillemot, C., and Fuchs, J.-J. (2008). Sparse approximations for joint source-channel coding. In *2008 IEEE 10th Workshop on Multimedia Signal Processing*.

Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S., Lugosch, L., Subakan, C., Dawalatabad, N., Heba, A., Zhong, J., et al. (2021). Speechbrain: A general-purpose speech toolkit. *arXiv preprint arXiv:2106.04624*.

Riesenhuber, M. and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature neuroscience*.

Rifai, S., Mesnil, G., Vincent, P., Muller, X., Bengio, Y., Dauphin, Y., and Glorot, X. (2011). Higher order contractive auto-encoder. In *Joint European conference on machine learning and knowledge discovery in databases*. Springer.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Rumelhart, D. E. and Zipser, D. (1985). Feature discovery by competitive learning. *Cognitive science*.

Sainath, T. N., Ramabhadran, B., Picheny, M., Nahamoo, D., and Kanevsky, D. (2011). Exemplar-based sparse representation features: From timit to lvcsr. *IEEE Transactions on Audio, Speech, and Language Processing*.

Sak, H., Senior, A. W., and Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling.

Shon, S., Mun, S., Kim, W., and Ko, H. (2017). Autoencoder based domain adaptation for speaker recognition under insufficient channel information. *arXiv preprint arXiv:1708.01227*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*.

Srivastava, R. K., Masci, J., Kazerounian, S., Gomez, F., and Schmidhuber, J. (2013). Compete to compute. *Advances in neural information processing systems*.

Stevens, K. N. (2000). *Acoustic phonetics*. MIT press.

Tang, H., Hsu, W.-N., Grondin, F., and Glass, J. (2018). A study of enhancement, augmentation, and autoencoder methods for domain adaptation in distant speech recognition. *arXiv preprint arXiv:1806.04841*.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*.

Varga, A. and Steeneken, H. J. (1993). Assessment for automatic speech recognition noisex-92: A database and an experiment to study the effect of additive noise on speech recognition systems. *Speech communication*.

Vendetti, J. (2002a). Lecture notes in phonetic transcription. *CS4705 Lecture notes*.

Vendetti, J. (2002b). Lecture notes in phonetic transcription. *CS4705 Lecture notes*.

Veselỳ, K., Ghoshal, A., Burget, L., and Povey, D. (2013). Sequence-discriminative training of deep neural networks. In *Interspeech*.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *International Conference on Machine Learning (ICML)*.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*.

Xiong, F., Barker, J., and Christensen, H. (2019). Phonetic analysis of dysarthric speech tempo and applications to robust personalised dysarthric speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Young, S. J., Odell, J. J., and Woodland, P. C. (1994). Tree-based state tying for high accuracy modelling.

Zhang, L., Lu, Y., Wang, B., Li, F., and Zhang, Z. (2018). Sparse auto-encoder with smoothed $l1$ regularization. *Neural Processing Letters*.

# Selen Hande Kabil

hande.kabil@gmail.com • **in** hande-kabil

## Education

**École Polytechnique Fédérale de Lausanne (EPFL)** 2018 - 2022
*Doctoral Student in Electrical Engineering*

**École Polytechnique Fédérale de Lausanne (EPFL)** 2015 - 2018
*Masters in Computer Science*

**Middle East Technical University (METU), Turkey** 2010 - 2015
*Bachelors in Computer Engineering*

## Work Experience

- **Idiap Research Institute** **Switzerland**
  *Research Assistant in Speech & Audio Processing Group* *May 2018 – Oct. 2022*
  **Supervisor:** Prof. Hervé Bourlard
  **Thesis title:** Sparse Autoencoders for Speech Modeling and Recognition
  - Designed various sparse autoencoder models for sparse modeling of speech
  - Improved performance over strong chain model baseline system for robust speech recognition on far-field speech
  - Repurposed sparse models in transfer learning framework for pathological speech recognition

- **Idiap Research Institute** **Switzerland**
  *Research Intern in Speech & Audio Processing Group* *Sept. 2017 – Feb. 2018*
  **Supervisor:** Dr. Mathew Magimai-Doss
  **Thesis title:** Understanding End-to-End Acoustic Modeling for Speech Recognition: A Case Study on Children Speech
  - Implemented end-to-end CNN-based acoustic models for children speech

- **Katia SA** **Switzerland**
  *Summer Intern* *July 2017 – Sept. 2017*
  - Developed spam call filter with n-gram language models and fusion of classifiers

- **Idiap Research Institute** **Switzerland**
  *Project Student* *Feb. 2017 – June 2017*
  **Supervisor:** Dr. Mathew Magimai-Doss
  - Implemented CNN-based gender classifier, which jointly handles feature extraction and classification given the raw speech signal

- **Hacettepe University Vision Lab** **Turkey**
  *Research Assistant* *July 2014 – June 2015*
  **Supervisor:** Dr. Ruken Çakici
  - Built a Turkish caption generation tool for images using n-gram language models

- **METU** **Turkey**
  *Project Student* *Sept. 2014 – Jan. 2015*
  - Developed an aspect-based opinion mining module for an integrated hotel management system as my senior design project

## Computer Skills

- **Programming Languages:** Python, C++, Bash
- **Toolkits:** Kaldi, Pytorch, numpy, matplotlib, scikit-learn, NLTK, pandas
- **Software Engineering:** Agile, Scrum, Version Control, Requirements Analysis

## Publications

- Bourlard, H., and Kabil, S.H. *Autoencoders reloaded*. Biological Cybernetics, 2022
- Kabil, S.H., and Bourlard, H. *From Undercomplete to Sparse Overcomplete Autoencoders to Improve LF-MMI Speech Recognition*. Interspeech 2022
- Dubagunta, S.P., Kabil, S.H. and Magimai-Doss M. *Improving Children Speech Recognition through Feature Learning from Raw Speech Signal*. ICASSP 2019
- Kabil, S.H., Muckenhirn H. and Magimai-Doss M. *On Learning to Classify Genders from Raw Speech Signal*. Interspeech 2018

## Research Reports

- Kabil, S.H., and Bourlard, H. *Speech Modeling using Sparse Autoencoders*. Idiap technical report. 2022
- Kabil, S.H., and Bourlard, H. *Sparse Autoencoders to Enhance Speech Recognition*. Idiap technical report. 2022

## Languages

- **English:** Fluent
- **German:** Intermediate
- **French:** Basic
- **Turkish:** Native